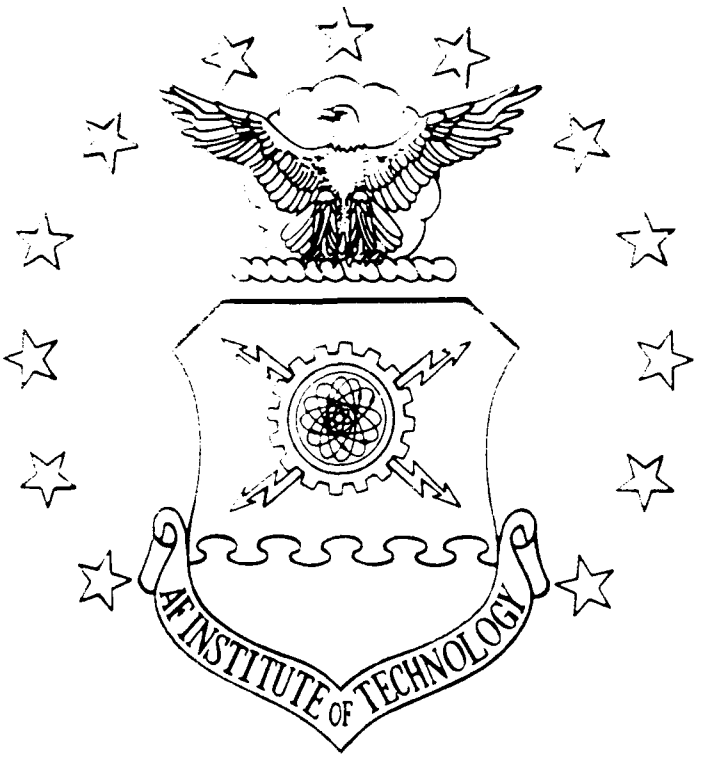


INVESTIGATION OF DIRECT AND INDIRECT OPTIMIZATION ALGORITHMS FOR AEROSPACE STRUCTURES

THESIS

Harry Hopkins III, Captain, USAF

AFIT/GA/ENY/90D-7

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

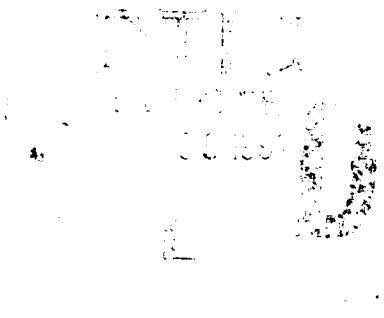
AFIT/GA/ENY/90D-7

INVESTIGATION OF DIRECT AND INDIRECT OPTIMIZATION ALGORITHMS FOR AEROSPACE STRUCTURES

THESIS

Harry Hopkins III, Captain, USAF

AFIT/GA/ENY/90D-7



Approved for public release; distribution unlimited

AFIT/GA/ENY/90D-7

INVESTIGATION OF DIRECT AND INDIRECT OPTIMIZATION ALGORITHMS FOR AEROSPACE STRUCTURES

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Harry Hopkins III, B.S.

Captain, USAF

December 1990

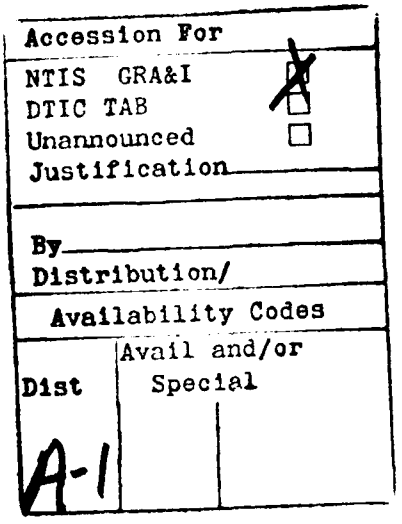
Approved for public release; distribution unlimited

Preface

This study investigates the performance of several methods of structural optimization. Three of the methods were programmed by myself along with a gradient calculator, which when linked with the finite element code FRAME form a structural optimization system.

The study is only a start in the comparison of Direct and Indirect methods, but the design system I developed can be further utilized to develop this study more rigorously.

I would like to thank Dr. Gans, Dr. Palazotto, and Dr. Spenny for their continuing patience, confidence and assistance in my work.



Harry Hopkins III

Table of Contents

Table of Contents

(Continued)

List of Figures

List of Figures

(Continued)

List of Figures

(Continued)

List of Tables

Abstract

This study investigated the performance of two direct and two indirect methods of structural optimization. A gradient calculator and overall design system was created with the finite element code FRAME. Three of the methods were employed using this system with the fourth being a commercial available code. The algorithms were applied to two different sized trusses using static constraints and were measured for accuracy, reliability, and efficiency. Results for the problems tested showed the Direct methods were sensitive to starting designs and their performance depended on the proper selection of internal parameters. They were also shown to have a high degree of accuracy and the flexibility to handle different problems. The Indirect methods showed that they were very effective when applied to specific problems and were simpler to implement and manage than direct methods, but lacked the flexibility to handle a variety of problems.

INVESTIGATION OF DIRECT AND INDIRECT OPTIMIZATION ALGORITHMS FOR AEROSPACE STRUCTURES

I. INTRODUCTION

BACKGROUND

The field of structural optimization is currently undergoing rapid changes in methods and focus. In recent years the application of optimization methods has become widespread in the aerospace industry. The methods and algorithms that will be utilized most in coming years will be those that work well, are user friendly and provide warning and assistance when algorithm failures and misuse occur. Although the past has produced several original Structural Optimization methods, the future will focus on only a few reliable and robust techniques.

There are two broad categories of Structural Optimization techniques currently available. They are the Indirect or Optimality Criteria methods and the Direct or Mathematical Programming methods. Optimality criteria are the conditions a structural design must satisfy at optimal design. This idea is central to the Indirect methods. The

Direct methods search the design space iteratively for optimum designs satisfying a convergent criteria.

The iterative approach is applied to both methods but in different ways. In the Direct method the mathematical programming involves three basic steps: selecting a starting design, determination of the travel direction in the design space, and determination of the appropriate step size in that direction. Experience ar' trial and error can give a good initial design. The determination of a travel direction involves the computation of the gradients of the constraint and objective functions and their transformations. These can cost in terms of computations. Selection of a step size is an art and can involve many trails. The computational cost becomes very important in the optimization of large scale structural systems where each iteration may require an analysis of the whole structural system.

The application of Indirect methods also involves three basic steps. A starting design must also be selected. The second step is to derive the necessary optimality criteria for the specific problem (the stopping point) and finally a efficient iterative algorithm to achieve this criteria must be selected (4:208).

The two major types of Indirect methods are Physical or intuitive based algorithms and Mathematical methods. Physical based algorithms employ derived explicit recurrence relations for redesign based on approximate expressions of

the constraints in terms of the design variables. These expressions are exact for determinate structures. Mathematical Indirect methods are based on the Kuhn-Tucker necessary conditions of optimality. These conditions form a set of nonlinear equations which are solved iteratively (1:362).

The Direct methods are sometimes classified into two categories: Primal methods and Transformation methods. Primal methods consider the problem's constraints explicitly while the transformation methods transforms the original constrained problem into a sequence of unconstrained problems. (2:1586)

The Direct and Indirect approaches (see Table 01) come from two philosophically different viewpoints. Knowing which approach to apply to what type of problem is important. An understanding of the features, limitations, and advantages of these two approaches will give insights into their practical applications (1:79).

BASIC DESIGN MODEL

Optimization, from an engineering stand point, is a technique of compromise and balance. It is applied to a system in the hope of reaching the systems most effective balance between its desired operating characteristics and its operating environment. In Structural Optimization the problem is further complicated by the structural analysis procedure which is most often the Finite Element (FE) method. "FRAME" and MSC/NASTRAN were the FE codes used in

this study. FRAME's implementation is discussed in the "Design System" section. MSC/NASTRAN was used with its own structural optimization routine. This section will discuss some basic Structural Optimization concepts and formulate the basic design model.

MATHEMATICAL PROGRAMMING METHODS (DIRECT)	
PRIMAL	**TRANSFORMATION**
*Sequential Quadratic Programming *Gradient Projection *Reduced Gradient *Feasible Directions *Sequential Linear Programming Projection Methods	*Sequential Unconstrained Minimization (Penalty and Barrier functions) *Multiplier (or Augmented Lagrangian)
OPTIMALITY CRITERIA METHODS (INDIRECT)	
PHYSICAL	**MATHEMATICAL**
*Fully Stressed Design	*Iterative Procedures based on application of Kuhn-Tucker conditions *Dual Methods

Table 1
Some Examples of Different Structural Optimization Methods

The minimum weight design of a structure is presented in the general form of a mathematical programming problem as follows:

$$Minimize \quad F(x) \tag{1}$$

$$Subject \ to:$$

$$g(x)_j \leq 0 \quad j = 1, m \tag{2}$$

$$x_i^l \leq x_i \leq x_i^u \quad i = 1, n \tag{3}$$

$F(x)$ is the objective function. This is most often the weight of the structure to be minimized. x is a vector of independent design variables. The design variables used throughout out this study are the member cross-sectional areas. The inequality constraints of eq. (2), sometimes called behavioral constraints, must be satisfied for a feasible design. Stress constraints are used in this study. The constraints of eq. (3) are side constraints. They act as upper and lower bounds on the design variables.

The objective function or the structures weight is formalized as follows:

$$Weight = \sum_{i=1}^{NE} \rho A_i L_i \tag{4}$$

where ρ is the material density, which will be constant for the whole structure. A_i is the cross-sectional area and L_i

is the length with NE being the number of elements or members of the structure. Some of the areas could be the same for different elements. This is called design variable linking and it reduces the number of design variables for the structure.

The stress constraints are normalized as follows:

$$\frac{\sigma_i}{\bar{\sigma}^-} - 1 \leq 0 \quad i=1,NE \tag{5}$$

$$\frac{\sigma_i}{\bar{\sigma}^+} - 1 \leq 0 \quad i=1,NE \tag{6}$$

where σ is the stress in member i. $\bar{\sigma}^-$ is the lower bound on the stress (maximum compressive stress) and $\bar{\sigma}^+$ is the upper bound on the stress (maximum Tensile stress).

Constraints for displacements can be imposed at prescribed joints and are normalized as follows:

$$\frac{u_{ij}}{\bar{u}_{ij}^-} - 1 \leq 0 \tag{7}$$

$$\frac{u_{ij}}{\bar{u}_{ij}^+} - 1 \leq 0 \tag{8}$$

where $\bar{u}_{ij}^+$ and $\bar{u}_{ij}^-$ are the upper and lower bounds on the displacements at the i th joint in the coordinate direction j.

Some important observations of the above standard design model are:

(1) The objective function and the constraints must depend on the design variables (cross-sectional area). If they aren't then they are not valid for this design problem.

(2) For a given optimum design, usually not all the inequality constraints will be satisfied at equality.

(3) While the objective function is linear in the design variables, the constraints are not. This causes the above problem to be classified as a non-linear programming problem (3:46-47).

(4) For almost all structural problems no explicit expressions can be written for the behavioral constraints (stress and displacements) in terms of the design variables. This leads to a numerical methods strategy which involves iterative approaches in search of the optimum design variables (4:208).

This basic design model will be used as a basic formulation for most of the methods discussed and will be expanded on in the forthcoming sections.

PROBLEM

The objective of this study is twofold. It provides documented procedures for using the FE code FRAME as the structural analysis tool for structural optimization testing and compares and contrasts some Direct and Indirect algorithms used in the optimization of planer trusses. The test problems used are formed using the static constraint of stress with a minimum weight objective. The optimization will be limited to sizing the structural members of a fixed configuration. The results will yield limitations and advantages of the two approaches as applied to small scale

planer truss design.

SUMMARY OF CURRENT KNOWLEDGE

The current research into Structural Optimization methods takes on many forms and directions. The novice can easily become lost. An attempt will be made to put some kind of framework on this research and document some very helpful sources for this study.

The "Approximation concepts approach" (5:1-45) in figure 1 (5:38) shows the pieces of a Structural Optimization System and their interaction in the optimization process. This concept was used in developing the system for the implementation of the Constrained Steepest Descent (CSD) method used in this study (see Design System Section).

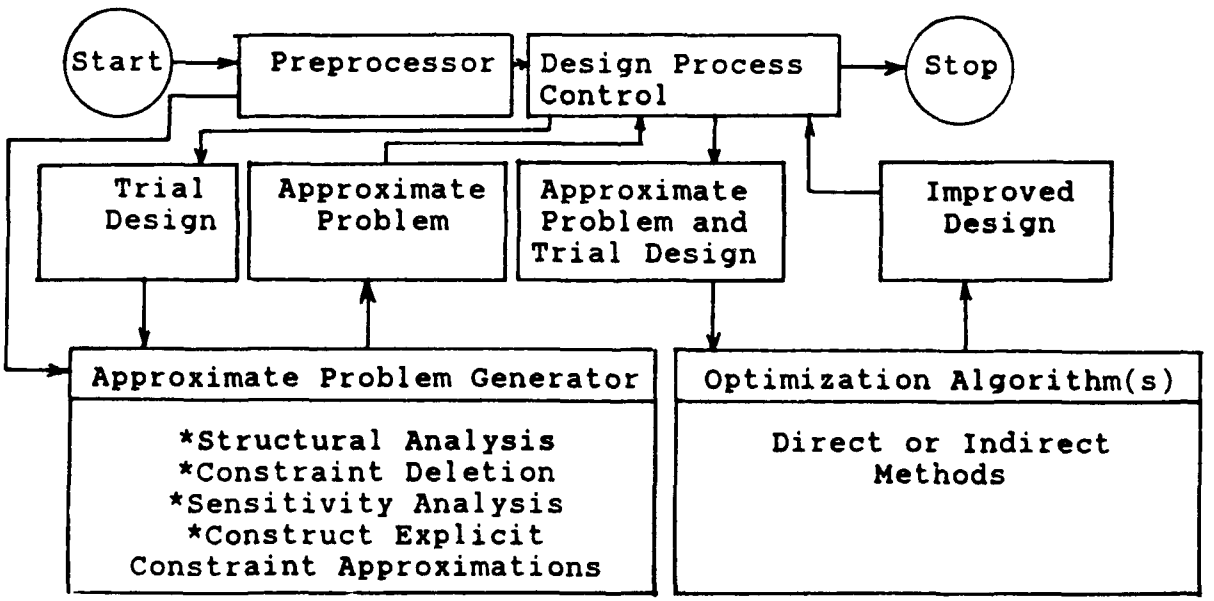


Figure 1. Approximation Concepts Approach

Figure 1 provides not only guidelines for constructing an optimization system but also a useful framework for putting the current research in Structural Optimization into perspective. The Figure shows that there are two major parts to most optimization systems. They are the Approximate Problem Generator and the Optimization Algorithm(s). The generator reduces the problem to a manageable one for the selected optimization algorithm. Much research involves looking into the approximate problem formulation.

The first step in the approximate problem formulation is to analyze the structure for a response to the current design variables. This can take place many times in an optimization procedure and may be computationally expensive. Active research in this area looks at approximate methods for the reanalysis. The goal is to improve computational efficiency will holding accuracy loss to a minimal. The next step is to take the results of the analysis and form an approximate problem that is manageable for the selected optimization algorithm. The problem is most often approximated by the use of sensitivity analysis data. These are the gradients of the objective function and the constraints with respect to the design variables. These calculations are also very costly and ways have been developed to improve their accuracy and reduce their cost. The calculations are most often used in a Taylor Series expansion of the objective function and the constraints to form a linear subprob-

lem for the optimizer.

Research into a structural optimization algorithms can also take many forms. The work can involve developing new procedures or adapting general mathematical optimization to structures. Currently much effort is going into testing existing algorithms. There are some problems to over come in this effort. The performance of an algorithm is not only dependent on it's procedure but also on the methods used in the rest of the optimization system like the forming of the subproblem. Performance of any algorithm is dependent on its implementation, coding and hardware used. The next question is what exactly is measured and analyzed in testing Structural Optimization algorithms.

The analysis of a Structural Optimization algorithm must be done in terms of Structural Optimization and not general mathematical optimization. This is done by considering some basic problems in the optimization of structures. In the formulation of the basic design problem the constraints are not only nonlinear but implicit in the design variables. Another concern is that the problem may be highly nonlinear and nonconvex. This nonconvexity means a problem may have many local minima which could make it more difficult to locate the global minima. The problems can also be very large, with 50 design variables and 500 constraint equations not uncommon (2:154).

These problems help form a basis for the comparative

study of Structural Optimization Algorithms. The analysis should have both an analytic study and numerical tests. The analytic study should examine the algorithms to see if they are especially suited for handling implicit functions. For Direct methods an examination of the geometrical significance of the search direction would show how it stands up to the nonconvex problem. The global convergence properties of the specific algorithm also need to be studied. Good global convergent properties mean a solution will converge from any starting point. This is an indication of the algorithm's robustness and reliability. The numerical tests will apply the algorithm to specific problems. These tests will show accuracy, reliability, and efficiency (2:158). The optimization system used in the testing also needs to be considered with its effects characterized and normalized as much as possible.

A survey of the work in Structural Optimization testing shows many tests and studies in the past, but none of the work uncovered revealed a direct comparison study between Direct and Indirect methods. Comparative studies of these algorithms were done within the Direct and Indirect categories.

Belegundu and Arora performed an extensive study of Direct methods (6:1619-1620). They compared eight different structural optimization codes applied to 12 different structural optimization problems. Three of these codes performed

well in accuracy, reliability, and efficiency when applied to small trusses. Large structures of 200 members gave poor results in accuracy. Two of the codes were based on multiplier algorithms; the other was base on an exterior quadratic penalty function technique.

An investigation of Indirect or Optimality Criteria Algorithms by Knot, Berke and Venkayya (7:182-189) examined a number of Mathematical methods. They showed that the algorithms differed only in the degree of approximations made in formulating their recurrence relations to modify the design variables and to evaluate the Lagrange multipliers. They also showed that these type of algorithms depend on the proper step size used in the recurrence relations.

Reklaits, Ravindran and Ragsdell in their book Engineering Optimization Method and Applications, present a whole chapter on comparison philosophy of constrained optimization methods with documentation of many large scale tests. The importance of robustness, ease of use and storage requirements as measures of merit for each algorithm are discussed. The chapter also points out the differences between algorithm and software testing. When an algorithm is coded and turned into software other factors need to considered such as coding efficiency and machine-dependent parameters,, which become part of any numerical testing. The large scale tests are of interest but are applied to many types of engineering design problems not just structural

design (8:533-535).

The problem of implementing an algorithm for structural design was studied by Tseng and Arora. The implementation of a Sequential Quadratic Programming (SQP) algorithm is examined in the context of it's performance. They develop the concept of numerical experiments and show how certain variations of the algorithm and it's parameters can effect it's performance. The conclusion is that effective implementation of a optimization algorithm requires expert knowledge and considerable numerical experimentation (9:1365).

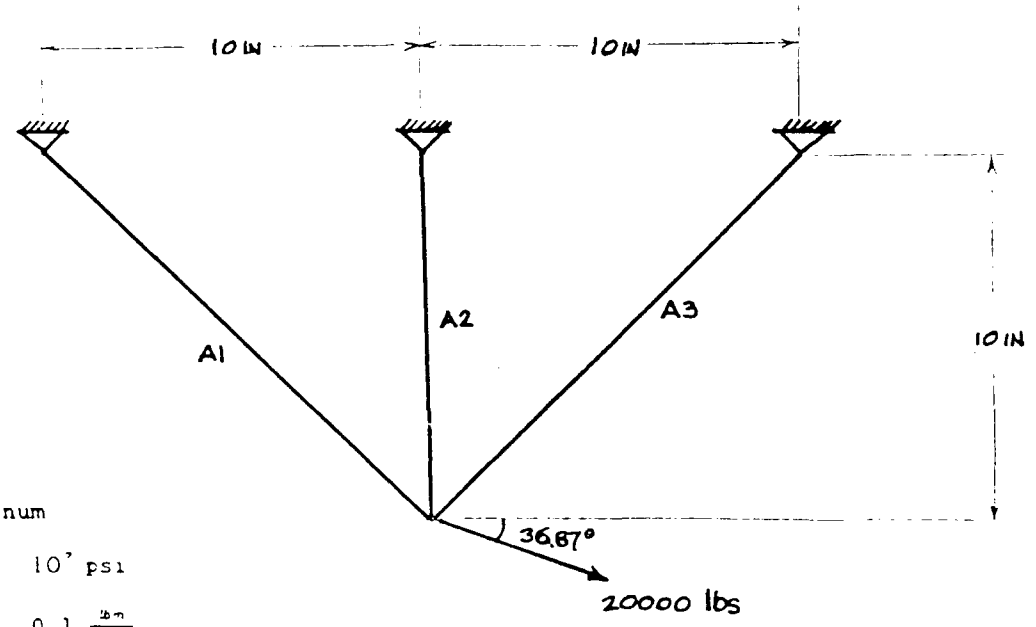
SCOPE

This study is limited to Structural Optimization problems of a fixed planer truss configuration. Shape or configuration optimization algorithms will not be looked at. The problems constraints will be limited to static stress constraints under one loading condition.

No testing will be done of large scale problems. A 3-bar and 10-bar planer trusses (figure 2 and 3) will be used as test problems.

STANDARDS

Accuracy, reliability, and efficiency are measured from all numerical tests. Accuracy is determined from the value of the objective function at final design. The algorithm which yields the minimum weight will be the most accurate. This solution is validated with other published sources. Reliability is measured from the algorithm's capability to



Material: aluminum

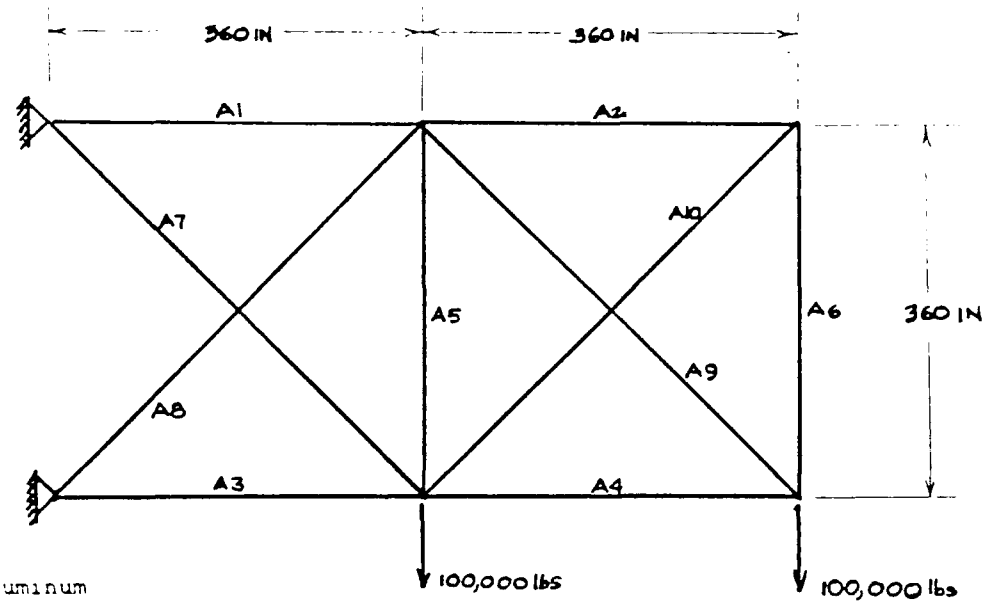
Young's Modulus: 10^7 psi

Specific Weight: 0.1 $\frac{lbm}{inches^3}$

Allowable Stress: +20000 psi
-15000 psi

Minimum Area: 0.1 $inches^2$

Figure 3. Three-Bar Truss



Material: aluminum

Young's Modulus: 10^7 psi

Specific Weight: 0.1 $\frac{lbm}{inches^3}$

Allowable Stress: +25000 psi
-25000 psi

Minimum Area: 0.1 $inches^2$

Figure 4. Ten-Bar Truss

deliver a feasible solution. A feasible solution need not be an exact optimum. For example an algorithm may give accurate solutions for some problems but may fail to give feasible solutions to others. Such an algorithm is unreliable. The algorithms efficiency is measured by the number of function calls and the total iterations needed for a solution.

APPROACH/METHODOLOGY

Phase 1: A First-Order Sequential Quadratic Programming (SQP) algorithm was selected from published sources. The algorithm is programmed along with a gradient calculator and fitted with the FE code FRAME (10:39-59) to form a Design System. A series of numerical experiments are run on the code. Problem runs are made with variations in program parameters and starting designs.

Phase 2: The Fully Stressed Design algorithm is programmed. Two different variations are tested and documented.

Phase 3: MSC/NASTRAN's method of Feasible Directions is tested.

Phase 4: The algorithms tested with data collected are analyzed and conclusions drawn.

MATERIALS AND EQUIPMENT

All computer codes were run on the Sun-4 Workstations. The study used the FE code FRAME and MSC/NASRTAN version 66.

II. DESIGN SYSTEM

BACKGROUND

An often ignored component of Structural Optimization testing is the Design System which supports the implementation of the algorithm. The Approximations Concept, shown in figure 1, provides the frame-work of a Design System. The basic strategy of the system "is to get the structural design optimization problem stated in equations (1) to (8) into a substitute form that adequately represents the actual design task and can also be handled by existing algorithms for efficient solution of explicit mathematical programming problems" (5:37). The Approximation concept achieves this by generating a sequence of relatively small, explicit, approximation problems that retain the essential features of the original design problem.

Following through one design iteration will present the details of the concept. Figure 1 shows that the first active block of the flow chart is a preprocessor. The preprocessor computes and stores all information that remains constant during the design iterations, like node coordinates for a truss system or material properties of the members. It also provides a trial design to the "approximate problem generator" through the "design process control" block. The generator applies a finite element structural analysis to the design. The resulting behavioral quantities (stress and displacements) are used to evaluate all the behavioral con-

straints. A constraint deletion technique is employed to temporarily ignore unimportant constraints and an active constraint set is formed. Sensitivity analysis is then applied to the active constraint set yielding a group of partial derivatives of these constraints with respect to the design variables. These derivatives are used to build approximate, but explicit, representations of the active set of constraints. The approximate problem is sent up to the "design process control" block again. There it is checked with the convergence criteria to see if the initial design is a solution. If not, the explicit objective solution is appended to the approximate problem, which now represents a tractable mathematical programming problem, and is passed to the optimization algorithm block.

The design is improved by the optimization algorithm block and passed back to the design process control block. Again the convergence criteria is checked and if it's not met the design becomes the starting design for the next iteration and is passed back to the approximate problem generator. It's important to note that the details of the Design System will be different with respect to the method or algorithm used. For example the stopping criteria for a Direct or Mathematical Programming method would normally be a diminishing returns criterion with respect to a further reduction in the objective function. For an Indirect or Optimality Criteria method this criteria might be the direct

satisfying of the Kuhn-Tucker conditions or some physical state selected to give an optimal design (5:40-41).

The emulation of the Approximation Concepts approach used in this study, employed the following published FORTRAN code listings: finite element code FRAME (10:359-361) and a code for the stiffness matrix inversion used in the gradient calculator (11:35-38). A users guide to the Design System is included in the Appendix.

THEORY

The two operations in the Design System that remain constant with respect to method or algorithm selected are the analysis (FRAME finite elements) and the sensitivity analysis or gradient calculator. Both these operations will be formalized in this section with their limitations and assumptions specified.

The initial problem formulation for finite element analysis includes member sizes, material properties (which may be different for each member), the configuration coordinates with specified support conditions and a set of external loads.

The analysis for the stresses and displacements must satisfy the conditions of equilibrium of forces at the nodes and compatibility of displacements. The analysis does not consider the weight of the individual members. The analysis includes the following assumptions: The trusses analyzed will be treated as discrete elements, with each element

treated as pin-connected with loads and reactions at the joints.

FRAME uses the Displacement method and considers the joint displacement components as the unknowns. The general case is defined as:

$$\underline{K}\tilde{u} = \tilde{P} \tag{9}$$

where $\underline{K}$ is the global stiffness matrix, $\tilde{P}$ is the vector of applied loads, and $\tilde{u}$ is the vector of displacements. An under line notes a matrix with a tilda noting a vector. After solving the displacements for every node the internal forces and stresses are calculated by applying the appropriate force relationships.

The gradient of the objective function or weight of the structure is very straight forward because of the linear relationship between the objective function and the design variables. The gradient of the displacement is formed by considering eqn (9) and taking the derivative of both sides with respect to the design variable $\tilde{X}_i$, $i = 1,2,\ldots,n$.

$$\frac{\partial \underline{K}}{\partial \tilde{X}_i}\tilde{U} + \underline{K}\frac{\partial \tilde{U}}{\partial \tilde{X}_i} = \frac{\partial \tilde{P}}{\partial \tilde{X}_i} \tag{10}$$

assuming the loads P are not a function of $\tilde{X}_i$

, and solving for $\frac{\partial \tilde{U}}{\partial \tilde{X}_i}$

, gives

$$\frac{\partial U}{\partial \bar{X}_i} = -\underline{K}^{-1}\frac{\partial \underline{K}}{\partial \bar{X}_i}U \tag{11}$$

and

$$\frac{\partial \underline{K}}{\partial \bar{X}_i} = \sum_{j=1}^{NE}\frac{\partial \underline{k}_j}{\partial \bar{X}_i} \tag{12}$$

with $\underline{k}_j$

being the element stiffness matrices and NE the no. of elements.

The derivative of the element stiffness matrix with respect to the design variable in eq (12) is very simple for the truss case where the matrix is the product of the design variable and a constant matrix.

The derivative of the stresses are handled as follows:

$$\frac{\partial \tilde{u}_j^o}{\partial \bar{X}_i} = \underline{T}\frac{\partial \tilde{u}_j}{\partial \bar{X}_i} \tag{13}$$

where $\frac{\partial \tilde{u}_j}{\partial \bar{X}_i}$ is a vector of partial derivatives of displacements of the nodes of element j with respect to a given design variable in the global frame. $\underline{T}$ is the transformation matrix for the global to the element frame and $\frac{\partial \tilde{u}_j^o}{\partial \bar{X}_i}$ is the transformed vector of partial derivatives of displacements for the nodes of element j with respect to design variable i in the elemental frame.

The equation for the force in a given member is

$$f_j = \frac{A_j E}{L_j}\tilde{u}_j^o \tag{14}$$

where $\bar{f}_j$ are the forces in element j, A_j is the cross sectional area of the member, E is Young's Modules for the structure, L_j is the length of the member, and $\bar{u}_j^o$ is a vector of the displacements for member j in the member frame. By taking the partial derivative of the element force equation (14) for the design variable and assuming the cross-sectional area of the element equals the design variable for this case. The following is obtained:

$$\frac{\partial \bar{f}_j}{\partial \bar{X}_i} = \frac{\bar{f}_j}{A_j} + \frac{A_j E}{L_j} \frac{\partial \bar{u}_j^o}{\partial \bar{X}_i} \tag{15}$$

and then finally

$$\frac{\partial \bar{\sigma}_j}{\partial \bar{X}_i} = \frac{\partial}{\partial \bar{X}_i} \frac{f_j}{A_j} \tag{16}$$

$$= \frac{1}{A_j} \frac{\partial \bar{f}_j}{\partial \bar{X}_i} - \frac{\bar{f}_j}{A_j^2} \tag{17}$$

Combining (16) and (17)

$$\frac{\partial \bar{\sigma}_j}{\partial \bar{X}_i} = \frac{E}{L_j} \frac{\partial \bar{u}_j^o}{\partial \bar{X}_i} \tag{18}$$

The two terms $\frac{\bar{f}_j}{A_j}$ and $-\frac{\bar{f}_j}{A_j^2}$ are non zero only if the element cross sectional area and the design variable are equal, but because they subtract out eqn (18) is for the general case. The gradients of the constraints are found by applying the above derivatives to eqns (5) to (8) (12:257-258).

The following set of gradients is the final result:

$$\frac{\partial F}{\partial \bar{X}_i} \quad i = 1,\ldots,NDV \tag{19}$$

and

$$\frac{\partial g_j}{\partial \bar{X}_i} \quad j = 1,\ldots,NCONST; \quad i = 1\ldots,NDV \tag{20}$$

where F is the objective function, g is the constraint, NDV is the number of design variables, and NCONST is the number of constraints. The gradient set is used in a Taylor series expansion of the objective function and active constraint set about the current design point to form an approximate problem for the Optimizer.

III. CONSTRAINED STEEPEST DESCENT METHOD (CSD)

BACKGROUND

A powerful class of modern Direct Methods are the Sequential Quadratic Programming (SQP) algorithms. These algorithms put the approximate problem into the form of a Quadratic Programming Problem. These methods have been investigated in many recent comparative studies and were found to be efficient methods for solving engineering optimization problems. Schititkowski found this method to require fewer function and gradient evaluations than many other methods (13:620-621). The Constrained Steepest Descent Method (CSD) is a very simple and basic interpretation of this class of algorithms.

The CSD algorithm is a first-order method for constrained optimization (3:384-389). In other words, It uses just first derivative information in forming the subproblem. It incorporates many of the features of Pshenichnyi's algorithm (9:1365-1366). The quadratic subproblem is formed using an identity matrix as the Hessian of the Lagrangian. The subproblem, with active constraints, is solved by forming its Kuhn-Tucker conditions. The resulting equations are solved using Phase I of the Simplex procedure. The Simplex procedure is used often in solving Linear optimization problems. After solving the subproblem Pshenichny's descent function along with an inexact line search is used to determine an improved design vector.

This section will develop the implementation of the CSD algorithm. The steps of the algorithm along with its theory and solution strategies are stated. Variations of the key conditions and parameters of the algorithm will be examined.

ALGORITHM AND THEORY

The Basic steps of CSD are as follows (3:392) (see figure 4):

Step 1. Set $k = 0$. The initial values for the design variable are specified as $\bar{x}^{(0)}$. An appropriate initial value for the penalty parameter R_0 is selected. Values are also specified for the constant γ,$(0 < \gamma < 1)$, δ the active constraint set parameter and two small numbers ϵ_1 and ϵ_2 which are the permissible constraint violation and convergence parameter, respectively. R and γ will be further defined in an upcoming step.

Step 2. The cost and constraint functions are evaluated at $\bar{x}^{(k)}$. The maximum constraint violation is defined as,

$$V_k = \max\{0; g_1, g_2, \ldots, g_m\} \qquad (21)$$

$$m = no.\ of\ constrants$$

g_m is the value of the *mth* constraint function.

Step 3. Define the active set of constraints as

$$I = \{i: g_j(\bar{x}^{(k)}) + \epsilon(\bar{x}^{(k)}) \geq 0; \quad j = 1 \ldots m\} \qquad (22)$$

with

$$\epsilon(\bar{x}^{(k)}) = \delta - V_k \qquad (23)$$

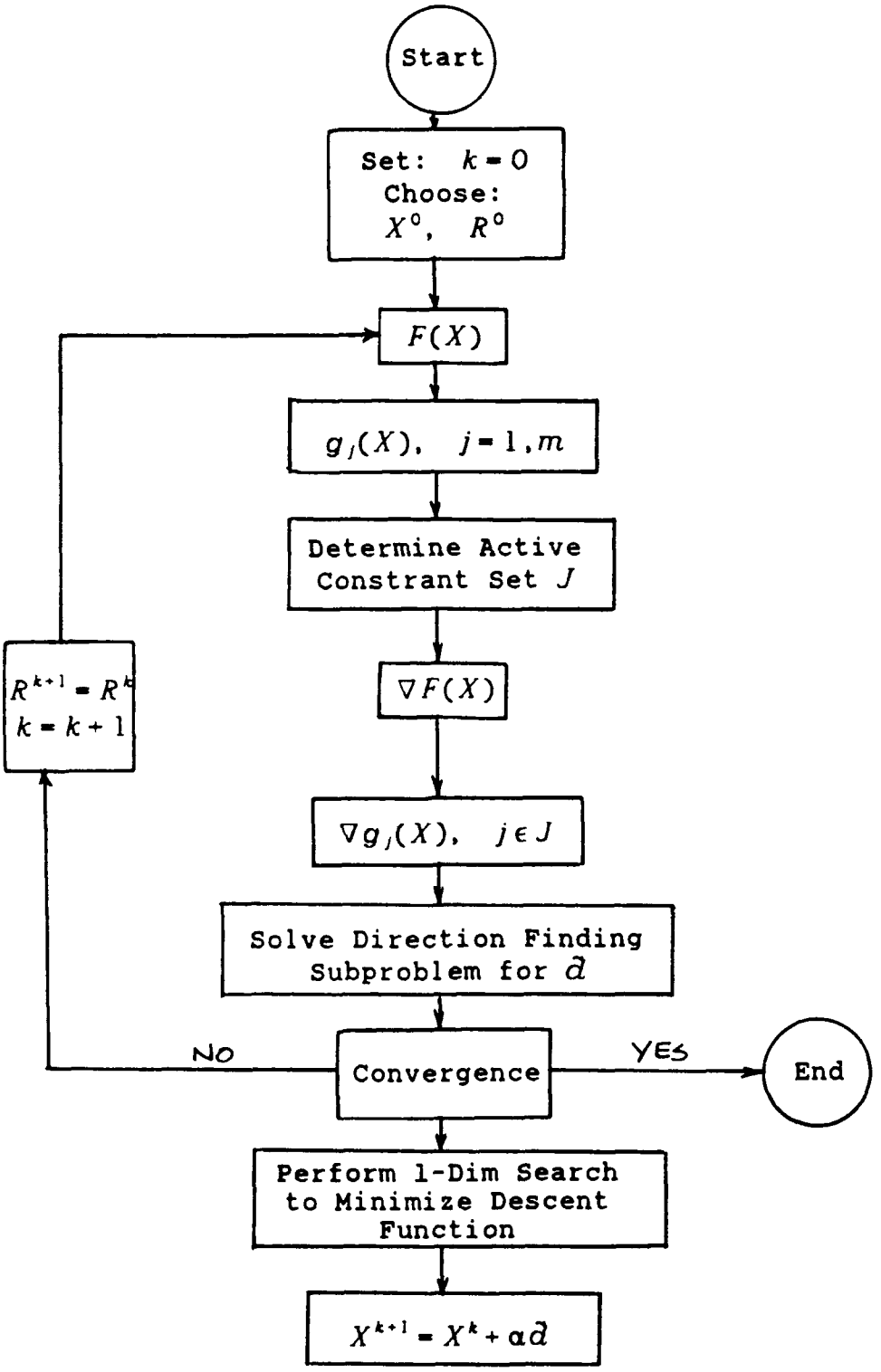


Figure 4. Constrained Steepest Descent Algorithm

Step 4. Calculate the gradients of the active constraint set. The gradients and the cost and constraint function values of the active constraints are formed into a QP sub-problem as follows,

minimize

$$q(\bar{d}) = \bar{c}^T\bar{d} + 0.5\bar{d}^T\underline{H}\bar{d} \tag{24}$$

subject to

$$\underline{A}^T\bar{d} \leq \bar{b} \tag{25}$$
$$\bar{d} \geq \bar{0}$$

where $\bar{c}$ = n dimensional vector of given constants and is defined as follows,

$$c_i = \frac{\partial f(\bar{x}^{(k)})}{\partial x_i} \quad (i = 1 \ldots n) \quad n = no.\ of\ design\ variables \tag{26}$$

or the *ith* component of the gradient of the cost function at the current design vector $\bar{x}^{(k)}$. $\bar{d}$ is an n dimensional design change vector or the search direction. It is defined as

$$d_i = \Delta x_i^{(k)};\ with\ (i = 1 \ldots n)\ n = no.\ of\ design\ variables \tag{27}$$

$\underline{H}$ is an $n \times n$ Hessian matrix of given constants. If $\underline{H}$ is positive definite, the QP problem is strictly convex and if a solution for the problem exists, it is a unique global solution. More advanced versions of this algorithm update this matrix, also called the Hessian of the Lagrange function, with first-order information with increased iterations. This study uses a constant identity matrix as $\underline{H}$. $\underline{A}$

is an $n \times m$ matrix of given constants and is composed of the following elements,

$$a_{ij} = \frac{\partial g_j(\bar{x}^{(k)})}{\partial x_i} \quad (i = 1 \ldots n) \quad n = no.\ of\ design\ variables \tag{28}$$

$(j = 1 \ldots m)$ $m = no.\ of\ inequality\ constraints\ within\ the\ active\ constrant\ set$

or the *ith* component of the gradient of the *jth* inequality constraint in the active constraint set at the current vector $\bar{x}^{(k)}$. $\bar{b}$ is an m dimensional vector with the following definition,

$$b_j = -g_j(\bar{x}^{(k)}) \tag{29}$$

$(j = 1 \ldots m)$ $m = no.\ of\ inequality\ constraints\ within\ the\ active\ constrant\ set$

or *jth* negative of the *jth* inequality constraint function in the active constraint set at the current design vector $\bar{x}^{(k)}$

This subproblem is then transformed into a form that can be solved with Phase I of the Simplex method. The Kuhn-Tucker necessary conditions are used to affect this transformation (3:378-379). To write the necessary conditions, slack variables $\bar{s}$ are introduced into the inequality constraints as follows,

$$\underline{A}^T \bar{d} + \bar{s} = \bar{b} \quad with \quad \bar{s} \geq 0 \tag{30}$$

The Lagrange function for the QP subproblem is defined as,

$$L = \bar{c}^T \bar{d} + 0.5 \bar{d}^T \underline{H} \bar{d} + \bar{u}^T (\underline{A}^T \bar{d} + \bar{s} - \bar{b}) - \bar{\xi}^T \bar{d} \tag{31}$$

where $\bar{u}$ and $\bar{\xi}$ are the Lagrange multiplier vectors for the inequality constraints and the nonnegativity constraints of Eqs. (25) respectively.

The Kuhn-Tucker necessary conditions are formed as follows,

$$\frac{\partial L}{\partial \bar{d}} = \bar{c} + \underline{H}\bar{d} + \underline{A}\bar{u} - \bar{\xi} = \bar{0} \qquad (32)$$

$$\underline{A}^T\bar{d} + \bar{s} - \bar{b} = \bar{0} \qquad (33)$$

$$u_i s_i = 0; \text{ with } s_i, u_i \geq 0 \qquad (34)$$

$(i = 1...m)$ m = *no. of inequality constraints within the active constant set*

$$\xi_i d_i = 0; \text{ with } \xi_i \geq 0 \qquad (35)$$

$(i = 1...n)$ n = *no. of design variables*

The resulting equations are then written into the following matrix form,

$$\underline{B} = \begin{pmatrix} \underline{H} & \underline{A} & -\underline{I}_{(n)} & \underline{0}_{(n\times m)} \\ \underline{A}^T & \underline{0}_{(m\times m)} & \underline{0}_{(m\times n)} & \underline{I}_{(m)} \end{pmatrix}_{(n+m)\times(2n+2m)} \qquad (36)$$

$$\bar{X} = \begin{Bmatrix} \bar{x} \\ \bar{u} \\ \bar{\xi} \\ \bar{s} \end{Bmatrix}_{(2n+2m)} \qquad \bar{D} = \begin{Bmatrix} -\bar{c} \\ \bar{b} \end{Bmatrix}_{(n+m)} \qquad (37)$$

By using these matrix definitions the Eqs. in (32) and (33) can be put into the following form,

$$\underline{B}\bar{X} = \bar{D} \qquad (38)$$

The switching and nonnegativity conditions of the Kuhn-Tucker necessary conditions in Eqs. (34) and (35) are enforced through the following relationship,

$$X_i X_{(n+m+i)} = 0; \text{ where } (i = 1...n+m) \qquad (39)$$

$$X_i \geq 0; \text{ where } (i = 1...2n+2m) \qquad (40)$$

In order to keep the search direction unrestricted and still meet the nonnegativity conditions, the search variables d_i are further decomposed into a positive and negative part.

The Phase I Simplex procedure used and programmed to solve the Eq. (38) system was deveioped by Wolfe (14:382-398) and refined by Hadley (15:370-375). The solution of the problem yields the new search direction vector $\bar{d}^{(k)}$ and Lagrange multipliers $\bar{u}^{(k)}$

Step 5. A check is made for the following convergence criteria,

$$||\bar{d}^{(k)}|| \leq \epsilon_2 \tag{41}$$

$$V_{(k)} \leq \epsilon_1 \tag{42}$$

If the convergence criteria is satisfied the process stops, otherwise it continues.

Step 6. The penalty parameter R is adjusted to meet the necessary condition $R \geq r_k$, where r_k is the sum of the Lagrange Multipliers from the QP subproblem defined as,

$$r_k = \sum_{i=1}^{m} u_i^{(k)} \tag{43}$$

Step 7. The new design vector $\bar{x}^{(k+1)}$ is then defined,

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} + \alpha_k \bar{d}^{(k)} \tag{44}$$

where α_k is the step size. The step size is found by an inexact line search. A sequence of trial step sizes,

$t_j = \left(\frac{1}{2}\right)^j$; $j = 0,1,2,3,4,\ldots$, are used to define a trial design point as

$$\tilde{x}^{(k+1,j)} = \tilde{x}^{(k)} + t_j \tilde{d}^{(k)} \tag{45}$$

The step size selected produces a trial design point that satisfies the descent condition. Pshenichny's descent function is used in forming this condition. Pshenichny's descent function Φ at the design vector $\tilde{x}$ is defined as,

$$\Phi(\tilde{x}) = f(\tilde{x}) + RV(\tilde{x}) \tag{46}$$

R and $V(\tilde{x})$ are the previous mentioned penalty parameter and maximum constraint violation respectively. The descent condition is defined as follows,

$$\Phi_{(k+1,j)} \leq \Phi_k - t_j \beta_k \tag{47}$$

where $\Phi_{(k+1,j)}$ is the descent function of Eq. (47) evaluated at trial step size t_j and corresponding design point $\tilde{x}^{(k+1,j)}$. β_k is defined as follows,

$$\beta_k = \gamma ||\tilde{d}^{(k)}||^2 \tag{48}$$

where γ is a constant between 0 and 1 that was specified in Step 1. The only variable in the decent condition Eq. (47) is t_j. A change in t_j causes a change in the trial design point which further changes the constraint function values. In this way a change in t_j affects both sides of the Descent condition Eq (47). The descent condition must be satisfied at each iteration to obtain a convergent algorithm.

Step 8. The current penalty parameter is saved as $R_{(k+1)} = R_k$

and the iteration counter is updated as $k = k + 1$. Control is then transferred to Step 2.

The CSD algorithm is convergent provided second derivatives of all the functions are piece-wise continuous (Lipschitz condition) (3:393) and the design vector $\bar{x}^{(k)}$ is bounded as follows:

$$\Phi(\bar{x}^{(k)}) \leq \Phi(\bar{x}^{(0)}); \quad k = 1,2,3,... \tag{49}$$

CSD COMPUTATIONAL CONSIDERATIONS

The effective implementation of an algorithm for structural optimization requires much experience. Considerable care and judgement are needed in the implementation of each algorithm step. Different implementation schemes can lead to different behaviors of the algorithm in terms of robustness and efficiency. This section will discuss and detail each step of the CSD algorithm.

Step 1 of the CSD algorithm starts with the selection of the parameters R_o, δ_o, and γ. A large R_o will keep the design vector close to the constraint boundary during the line search and minimization of the descent function Eq (46). This can slow down the convergence process (9:1369). A smaller R_o can lead to a larger step size in the first iteration which can change the history of the iteration process. For this study the value of R_o was varied over the values of 1, 5, 10, and 50 with the best result being compared to the other methods. Previous studies indicate

that $R = 1$ achieves good results (9:1369)

The function of γ forms the constant β in the second term of the descent condition Eqs (47) and (48). This term enforces a finite reduction in the descent function Eq (46) at every iteration. γ is selected as a positive number between 0 and 1. It's effect can be seen in figure 5.

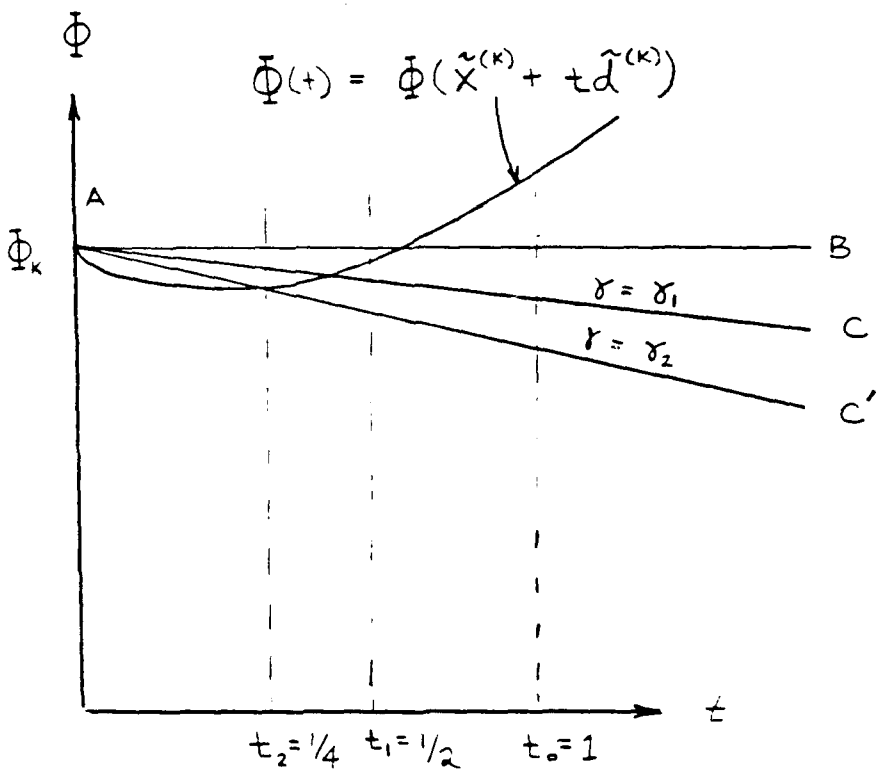


Figure 5. γ Effect on Step Size Selection

Here $\gamma_2 > \gamma_1$. γ directly controls the constant β_k which is the slope of the line $t\beta_k$. Line A-C is designated as $\gamma = \gamma_1$ and line A-C' as $\gamma = \gamma_2$. Figure 5 shows that a larger γ will tend to reduce the step size in order to satisfy the descent condition Eq (47). This can tend to slow down the convergence process (3:389). γ is set to 0.2 for this study. It has been shown to have good results (9:1370).

The value of δ_o determines the potential or active constraint set at each iteration. Larger values of δ_o will enlarge the potential constraint set. This results in more gradients being calculated and a larger QP subproblem to solve. The larger the set of constraints in the QP subproblem, the greater the risk of an infeasible QP subproblem. This is the result of inconsistencies of the constraints in their linearized form. δ_o was set to a fixed value of 0.1 and $\epsilon(b)$ was required to be non-negative. This was used with good results in other studies (9:1370).

The initial design vector also has an affect on the convergence performance of this algorithm. A bad starting point with a high percentage of violations will enlarge the potential constraint set which will lead to a large amount of gradient calculations and the risk of an infeasible QP subproblem. The resulting search vector may also be poor which could lead to convergence problems. Several starting designs are used in both test problems.

The next step is the forming and the solving of the QP

subproblem in step 4. The definition of the active or potential constraint set can affect the efficiency and quality of solution in three ways: 1) the number of gradient evaluations, 2) the dimension of the QP subproblem, and 3) the quality of the search direction (9:1370). The specified constant δ is used in Eq (23) to yield $\epsilon(b)$ which is used in Eq (22) to determine the potential constraint set. In Eq. (23), if $F(b) > \delta$, then $\epsilon(b)$ is a negative number which could result in even the violated constraints not being included in the potential constraint set. This would yield a poor search direction with any improvement in constraint violation due to chance. The δ constant must be selected to balance the size of the QP subproblem (to reduce the number of gradient calculations and risk of an infeasible problem), against a effective potential constraint set (to yield a good search direction). Much of the research on versions of this algorithm (9:1370) have show the good results with $\delta = 0.10$ and $\epsilon(b)$ required to be non-negative (17:2191). This study will use those specifications.

The QP subproblem solution process can also provide numerical difficulties. Although the subproblem is strictly convex and if a solution exists it is unique, the problem can be infeasible due to inconsistency of the linearized constraints. This problem has been encountered in many other studies and was difficult to deal with in this study.

In step 5 a check is made for convergence. More

iterations will be required if the criteria is stricter.

The penalty parameter R is adjusted in step 6 for preparation of step size selection. As pointed out above, a large value of R will force the design to remain close to the constraint boundary. Previous studies have showed that this has resulted in slower convergence with more steps and function evaluations (16:251). With R too small , other studies have also showed that the algorithm had a tendency to take large steps into the infeasible region when no new constraint was encountered. The method for R adjustment is kept constant.

The number of steps to allow in the algorithms attempt to minimize the decent condition is an important consideration in step 7. Most implementations of this algorithm do not specify how large j should be allowed to increase in $(0.5)^j$ for step size determination. A maximum of 25 steps is used here.

IV. FULLY STRESSED DESIGN (FSD)

BACKGROUND

The Fully Stressed Design (FSD) is one of the most successful Indirect Methods (Optimality Criteria Methods). It is similar to most Indirect Methods in that it's approach consists of an iterative application of analysis and a redesign rule or recurrence relationship. After each analysis if a member is found to be overstressed in the given load condition, the redesign rule increases the size of that member to reduce the stress or the opposite is done if the member is understressed. It's optimality criterion is as follows: For the optimum design each member of the structure is fully stressed at least under one of the design load conditions or is at it's minimum constraint size (18:196) The FSD method is applicable to structures which are subject only to stress and minimum member size constraints.

The FSD concept not only has intuitive appeal, but is significant for the following reasons (1:89):

1) Engineering experience indicates that a good design is often one in which each member is subjected to its allowable stress.

2) FSD can be proved to be a minimal weight optimal under certain circumstances.

3) FSD procedures are relatively efficient in comparison with many mathematical programming (Direct) methods.

4) An FSD is often a good starting point for optimum design

procedures based on mathematical programming (Direct) methods.

The FSD method will behave differently for statically determinate and indeterminate structures. Its been proven that the FSD is optimal for a statically determinate structure under a single load condition (19:1) and will convergence in one iteration step. Modification of member sizes for statically indeterminate structures will change the force distribution to the members. This will cause a few cycles of iterative analysis and redesign to obtain a fully stressed design. In many structures the forces in the members are little affected by the relative variations in the size of the other members and the iterative process converges rapidly. For other structures where this may not be so the convergence might be slow. An FSD is also not unique for an indeterminate structure. There may be more then one FSD for an indeterminate structure with no guarantee that the given FSD is a minimum weight FSD. Also some structures may not have an FSD at all. Each result must be examined and if needed the Kuhn-Tucker conditions should be applied to check for optimum design. There is a large body of experience in the use of FSD and it does indicate that in many problems the resultant design is indeed either the optimum or close to it (1:90).

ALGORITHM AND THEORY

A first-order approximation is defined for stresses in the $(k+1)th$ iteration as follows,

$$\tilde{\sigma}^{(k+1)} \approx \tilde{\sigma}^{(k)} + \underline{\nabla \sigma_x^{(k)}}(\tilde{x}^{(k+1)} - x^{(k)}) \tag{50}$$

where $\tilde{x}^{(k)}$ is a vector of design variables taken to be the cross sectional areas of members, $\tilde{\sigma}^{(k)}$ is a vector of stresses for members, and $\underline{\nabla \sigma_x^{(k)}}$ is a matrix of stress gradients. All values are at the kth design point. In the FSD method the object is to find a new design for the allowable stresses $\tilde{\sigma}^U$. Eq. (50) now becomes

$$\tilde{\sigma}^U \approx \tilde{\sigma}^{(k)} + \underline{\nabla \sigma_x^{(k)}}(\tilde{x}^{(k+1)} - x^{(k)}) \tag{51}$$

This first-order approximation can pictured as a plane tangent to the exact constraint surface. It is possible to solve for $\tilde{x}^{(k+1)}$ by solving the set of Eqs. (51), with the following concerns (1:90):

1) Divergence may occur when $\tilde{x}^{(k)}$ is a poor estimate of the exact solution.

2) The stress constraints which are active at the optimum might not be know in advance.

3) Partial derivatives $\underline{\nabla \sigma_x^{(k)}}$ must be calculated.

4) This solution requires the solution of simultaneous linear equations.

In terms of change in a single design variable, x_i, Eq. (51) becomes

$$\tilde{\sigma}^{U} \approx \tilde{\sigma}^{(k)} + \frac{\partial \sigma(k)}{\partial x_i}(x_i^{(k+1)} - x_i^{(k)}) \tag{52}$$

$x_i^{(k+1)}$ can also be calculated for a specific σ_j where j indicates a specific member as follows

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\sigma_j^{U} - \sigma_j^{(k)}}{\left(\frac{\partial \sigma_j}{\partial x_i}\right)^{(k)}} \tag{53}$$

Now the member stress σ_j can be expressed as

$$\sigma_j = \frac{A_j}{x_j} = A_j Y_j \tag{54}$$

A_j is the member force and Y_j is the member compliance. The partial derivative of σ_j with respect to x_i is

$$\frac{\partial \sigma_j}{\partial x_i} = -\frac{A_j}{x_i^2}\delta_{ij} + \frac{1}{x_j}\frac{\partial A_j}{\partial x_i} = A_j \frac{\partial Y_j}{\partial x_i}\delta_{ij} + Y_j \frac{\partial A_j}{\partial x_i} \tag{55}$$

$$\delta_{ij} = \begin{cases} 1 & if \quad i = j \\ 0 & if \quad i \neq j \end{cases} \tag{56}$$

Eq. (54) shows that the change in stress of a member is equal to two parts. The first part is attributed to a change in member size at constant force, with the second part caused by a change in force distribution at constant member size. In a structure that is statically determinate, $\frac{\partial A_j}{\partial x_i} = \frac{\partial A_j}{\partial Y_i} = 0$ if $i \neq j$ then the change in σ_j can only be due to the force redistribution. Ignoring the redistribution of forces due to a change in x_i , $\frac{\partial A_j}{\partial x_i} = 0$ is assumed. The compliance is selected as the design variable, hence $x_i = Y_i$ and $\frac{\partial Y_i}{\partial x_i} = 1$. For

$i \neq j$ the stresses remain constant. With $i = j$ Eq. (55) for the *kth* iteration cycle becomes

$$\left(\frac{\partial \sigma_i}{\partial x_i}\right)^{(k)} = A_i^{(k)} \tag{57}$$

Substituting Eq. (57) into (53) for $i = j$ yields

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\sigma_i^U - \sigma_i^{(k)}}{A_i^{(k)}} \tag{58}$$

Making the following substitutions into Eq. (58): $Y_i^{(k)} = x_i^{(k)}$, $A_i^{(k)} Y_i^{(k)} = \sigma_i^{(k)}$, produces

$$Y_i^{(k+1)} = \frac{\sigma_i^U}{A_i^{(k)}} \tag{59}$$

If the design variables are selected again as $x_i = \frac{1}{Y_i}$ and $A_i^{(k)} = \sigma_i^{(k)} x_i^{(k)}$ is substituted into Eq. (59) the FSD stress ratio redesign rule is produced

$$x_i^{(k+1)} = x_i^{(k)} \frac{\sigma_i^{(k)}}{\sigma_i^U} \tag{60}$$

This equation is also know as the Zero-Order approximation (1:91). This is because the exact constraint surface is approximated by a plane normal to the *ith* axis.

The design procedure consists of a cyclic analysis of the structure in which the member sizes are increased or decreased by the ratio of computed stress to the allowable stress for the member. The new computed sizes are then used in the next analysis iteration. This is continued until a solution converges (see figure 6).

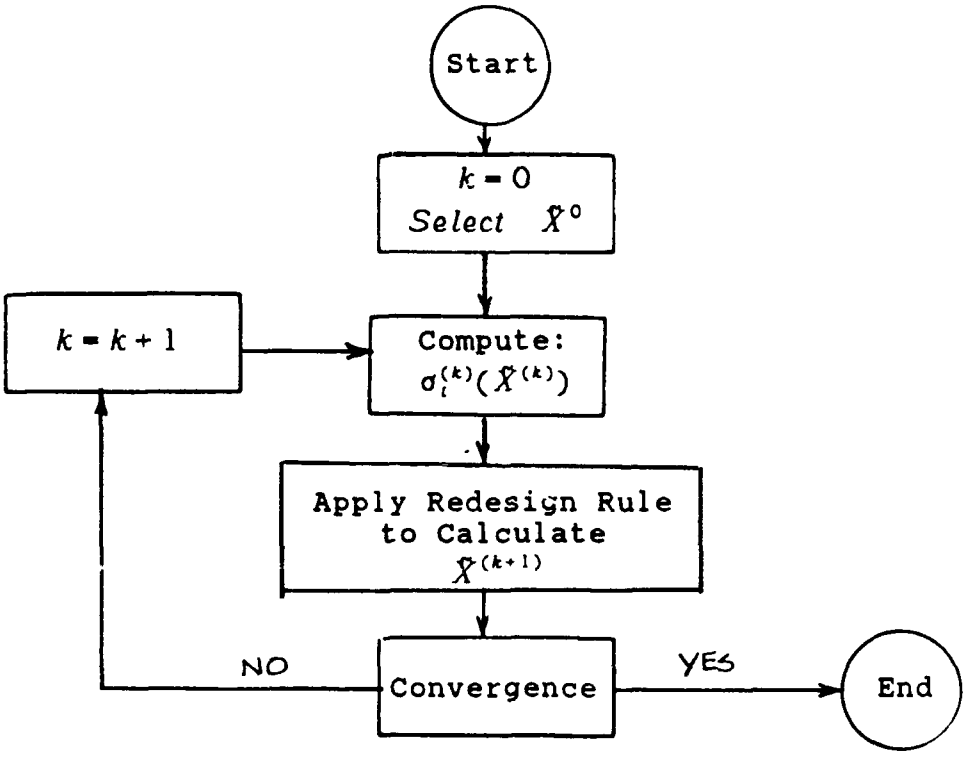


Figure 6. Fully Stressed Design Algorithm

A variation to FSD was proposed by Venkayya (20:1) with the following modified optimality criteria: the optimum design is the one in which the strain energy of each element bears a constant ratio to it's energy capacity. The energy capacity is defined as total strain energy stored if the element is stressed to it's limiting normal stress.

Energy capacity is independent of the actual state of stress in the element and depends only on the volume and on the limiting normal stress of the element. Energy capacity for an axial member of the *ith* element is

$$u_i^y = \frac{1}{2}\sigma_i^y \epsilon_i^y l_i x_i = \frac{1}{2E} l_i (\sigma_i^y)^2 x_i \qquad (61)$$

where σ_i^U and ϵ_i^U are the limiting normal stress and strain, respectively, l_i is member length, E is modules of elasticity, and a x_i is the cross-sectional area of the member. The strain energy of a member is

$$u_i = \frac{1}{2} A_i r_i = \frac{A_i^2 l_i}{2 E x_i} \tag{62}$$

A_i is again the axial force and r_i is the member displacement. Venkayya's optimality criteria stated that the strain energy of each member should bear a constant ration to its energy capacity, so taking the ratio of Eqs. (62) and (61) gives

$$\frac{u_i}{u_i^U} = \frac{A_i^2}{x_i^2 (\sigma_i^U)^2} = \frac{1}{V^2} \tag{63}$$

V is a constant of proportionality. Solving for x_i leads to

$$x_i = V \frac{A_i}{\sigma_i^U} \tag{64}$$

Transferring Eq. (64) into a redesign rule gives

$$x_i^{(k+1)} = V \frac{A_i^{(k)}}{\sigma_i^{(u)}} \tag{65}$$

$V = 1.0$ if $\frac{u_i}{u_i^U} = 1.0$ is assumed. This is equivalent to the stress-ration rule of Eq. (60). Because FSD and the optimum design are not always the same, a scaling scheme was also introduced by Venkayya. The design was rescaled after each redesign step. If the weight of the scaled design is larger of the weight of the previous cycle the process stops.

So a scalar Λ is defined which when changed will

adjust the absolute response of the structure. The following relationship is formed

$$\tilde{x} = \Lambda \bar{x} \tag{66}$$

$\tilde{x}$ is a vector of the actual design variables and $\bar{x}$ is a vector of relative design variables. Substituting Eq. (66) into Eq. (65) yields

$$(\Lambda \bar{x}_i)^{(k+1)} = V \frac{A_i^{(k)}}{\sigma_i^U} \tag{67}$$

The iterative procedure works much the same as the stress-ratio procedure. The procedure is easily viewed from figure 7 (1:94).

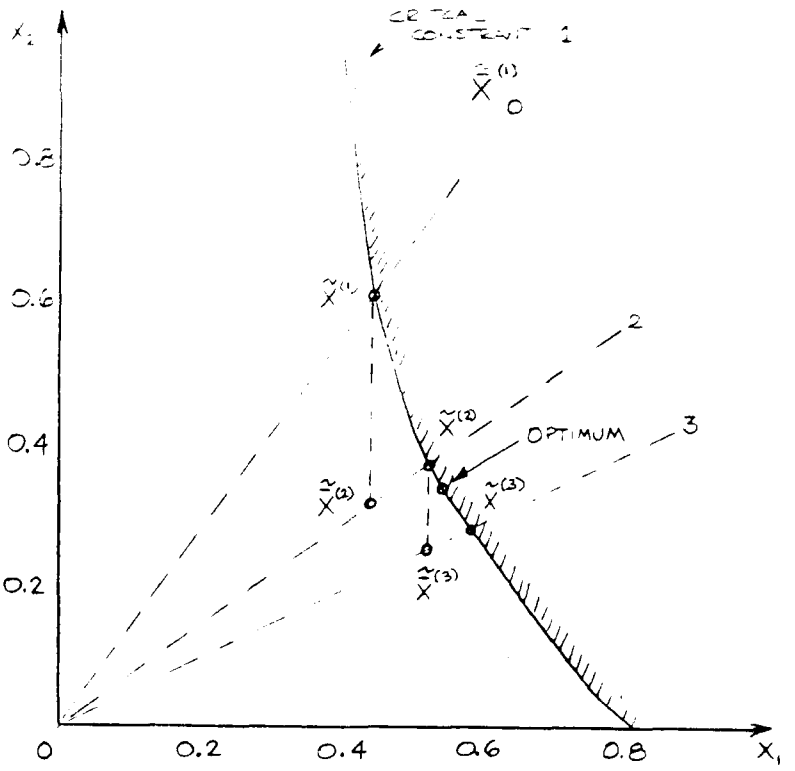


Figure 7. Design Space with FSD-Scaling

Figure 7 shows a design space of two variables. The lines 0-1,...,0-3 are called design lines. The relative design variable vector $\bar{x}$ is the same for every point on a given design line. The design moves up and down the line by scaling the design in other words the value is changed by the scalar Λ. Λ is computed to bring a design vector to the intersection of the constraint surface which is the lowest-weight feasible

design for that design line. The procedure works as follows:

1) The structure is first analyzed with the starting design or a relative design variable vector $\bar{x}^{(1)}$.

2) A corresponding scalar Λ to bring this relative point along the design line to the constraint surface $x^{(1)}$ is computed by $\Lambda = \frac{\bar{\sigma}}{\sigma^u}$, with $\bar{\sigma}$ being the stress at $\bar{x}^{(1)}$. The ratio of the scalar Λ is the largest value calculated for each constraint. This corresponds to most critical constraint of the design cycle.

3) $x^{(1)}$ is now calculated from $x^{(1)} = \Lambda \bar{x}^{(1)}$

4) Eq. (67) is then used to calculate a new relative design variable $\bar{x}^{(2)}$.

5) The weight is checked to see if it has reduced or increased. If is has increased the process is stopped and the design vector before the last cycle is selected as computed the design. If it has been reduced, control is then transferred to step 1 and a new cycle begins.

FSD COMPUTATIONAL CONSIDERATIONS

The FSD while successful with many problems and simple to use may have some difficulties. As indicated in tne previous section FSD may not always yield a true optimal design. The structure might have no FSD or more then one FSD. Their also might be no control to which FSD the structure converges to.

FSD with the strain-energy criteria will not always yield a true optimum (20:267). Where the iteration starts in the design space can greatly affect its search. It is also possible to miss the optimum between steps of the iteration process.

V. FEASIBLE DIRECTIONS ALGORITHM (FD)

BACKGROUND

The method of Feasible Directions (FD) is another powerful technique in the Direct Methods class. Like the Constrained Steepest Descent Method (CSD), it considers the constraints directly by solving a linerized subproblem for a search direction. A step size is then computed in that direction and the current design is updated.

The major difference between CSD and FD is the method in which the search direction is selected. For CSD a search direction is selected that will reduce the objective function and the constraint violations. The FD method picks the search direction to also reduce the objective function but it maintains a feasible design in its search.

This section will present a basic overview of the MSC/NASTRAN implementation of FD. The basic steps of the algorithm along with its theory and solution strategies are stated.

ALGORITHM AND THEORY

The basic steps of the FD algorithm (21:A11-A23) (see figure 8) are as follows:

Step 1. Set $k = 0$. The initial values for the design variables are specified as X^0.

Step 2. The cost and constraint functions are evaluated at X^k.

Step 3. The set of critical and near critical constraints

are defined.

Step 4. The gradients of the critical constraint set are calculated and formed into a subproblem as follows, minimize

$$g(\vec{d}) = \vec{c}^T\vec{d} \tag{68}$$

subject to

$$\underline{A}^T\vec{d} \leq 0 \tag{69}$$

$$\vec{d}^T\vec{d} \leq 1 \tag{70}$$

Where $\vec{d}$ is an n dimensional vector of search directions, and $\vec{c}^T$ is the gradient of the objective function. $\underline{A}^T$ is a matrix of gradients for the critical constraint set. The objective function in Eqn (68) provides the useability requirement. The minimization of this relationship provides a search direction that will reduce the objective function. Eqn (69) ensures the search direction takes place in the feasible region of the design space. The purpose of Eqn (70) is to prevent an unbounded solution. The Kuhn-Tucker conditions are applied to the subproblem resulting in a set of linear equations similar to those obtained in the CSD method. The solution to this set of equations results in a usable-feasible search direction. One side note is that if the initial design is infeasible the algorithm is modified to bring the design point back to the feasible region.

Step 5. A one-dimensional search is performed to obtain the step size in the search direction. This step is very

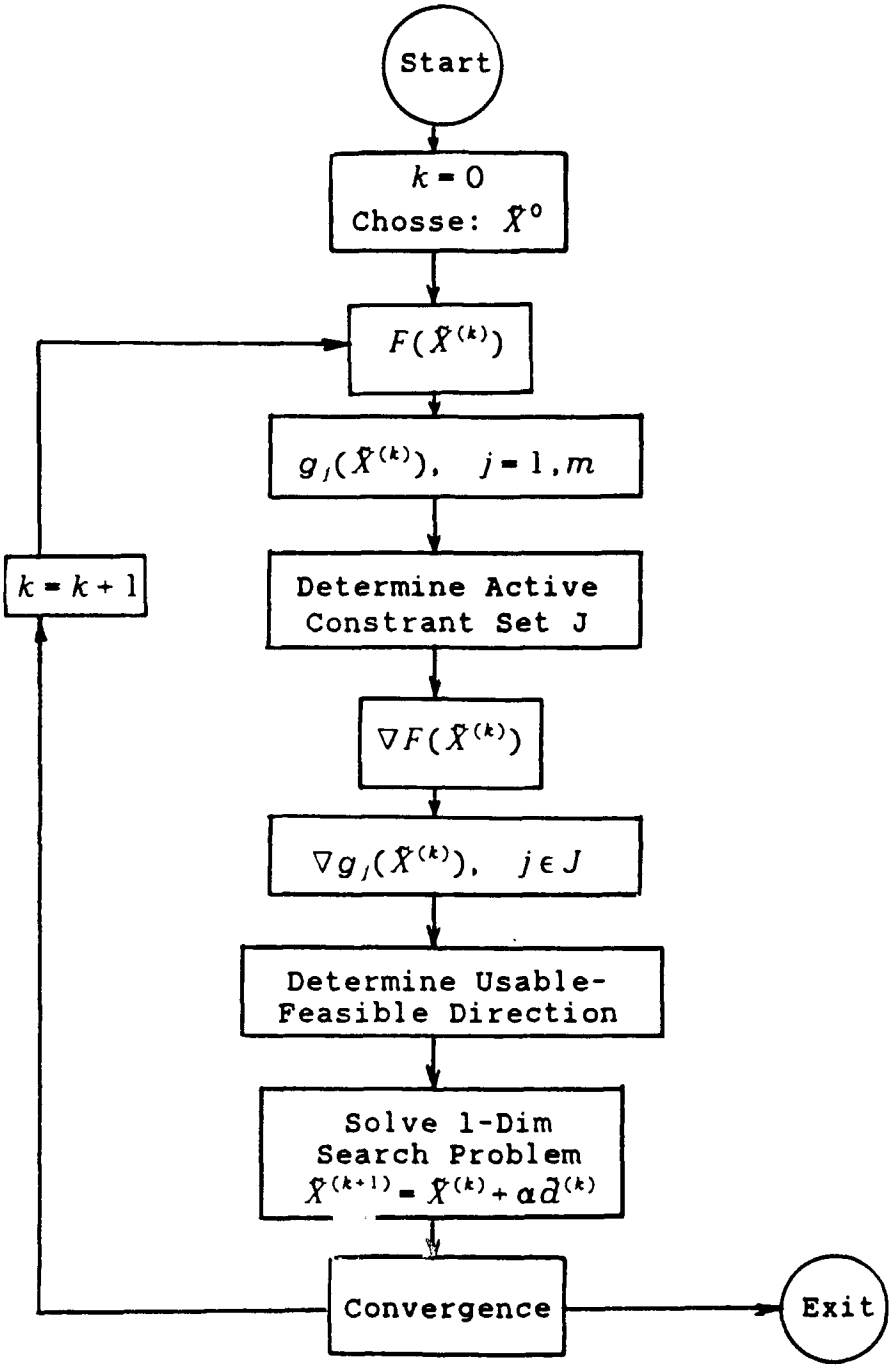


Figure 8. Feasible Direction Algorithm

different from the CSD method. Polynomial approximations to the objective and constraint functions are used as opposed to a full finite-element evaluation in CSD. Upper and lower bounds of the step size are determined and polynomial interpolation is used to refine the selected step.

Step 6. The new design point is checked for diminishing returns through the following relationship,

$$\frac{|F(\bar{x}^k)-F(\bar{x}^{k-1})|}{|F(\bar{x}^{k-1})|} \leq \epsilon_1 \tag{71}$$

where $F(\bar{x}^k)$ is the objective function at the *kth* iteration. ϵ_1 is taken to be .001 for this study. The absolute change in the objective function is also checked with,

$$|F(\bar{x}^k)-F(\bar{x}^{k-1})| \leq \epsilon_2 \tag{72}$$

with ϵ_2 equal to the maximum of $0.001|F(x^0)|$ and 0.0001. One of these criteria must be satisfied on 2 or more consecutive iterations.

FD COMPUTATIONAL CONSIDERATIONS

The MSC/NASTRAN structural optimization routine is a very refined and well tested commercial code that uses a modified form of the FD algorithm. Some of its strongest features are its approximation methods used in the step size selection. This holds to a minimum the number of finite element evaluations required. It also takes care of an infeasible starting design by bring the design back into the feasible region before implementing the normal FD algorithm.

VI. RESULTS

BACKGROUND

Testing took place in two phases. The first phase was a test of the CSD method with the maximum violation penalty parameter R taking on the values of 1, 5, 10, and 50. The best performance of CSD at a given R was used to represent the CSD method in the comparison of all four methods in the second phase. Because the accuracy of the final solutions of CSD for each of the parameters varied little for each given case, the most efficient (no. of function calls) parameter was selected as best.

All testing was done with the three and ten-bar truss problems, although CSD was not at all successful with its application to the ten-bar truss and results for that application are not considered. Reasons for this inability to converge to a solution are discussed later.

All testing was done from 10 different case starting conditions for the three bar problem and 5 different case starting conditions for the ten-bar application. Tables 2 and 3 give the weights and maximum violations for each case starting condition for the two problems. A variety of different starting conditions were used with varying degrees of over and under design. The optimal weight used as the test standard for the three-bar problem was 2.70 lbs (21:8.1-4) and 1593.2 lbs (18:237) for the ten-bar case.

CASE NO.	WEIGHT (LBS)	MAX CONSTRAINT VIOLATION
1	3.828	0
2	0.3828	6.414
3	28.384	0
4	38.284	0
5	10.282	6.503
6	3.328	0
7	166.421	0
8	0.6644	4.470
9	2.707	0.0048
10	64.396	0
OPTIMAL	2.70	0

Table 2. THREE-BAR TRUSS STARTING DESIGNS

CASE NO.	WEIGHT (lbs)	MAX CONSTRAINT VIOLATION
1	419.65	7.185
2	41.964	80.854
3	10512.15	93.963
4	20419.31	0
5	83929.2	0
OPTIMAL	1593.2	0

TABLE 3. TEN-BAR TRUSS STARTING DESIGNS

DISCUSSION OF PHASE I RESULTS

The test results for the CSD method, applied to the three-bar truss, are summarized in figures 9 to 18. The graphs show the design iteration history of the weight with respect to design cycles. Function calls are the number of finite element evaluation required. As all the figures show each design cycle might require a number of function calls. This is the result of the inexact line-search used by CSD which uses no approximation methods in the determination of a step size for a given the search direction.

For several cases and values of R the algorithm did not converge. The non-convergence of this method is attributed to a poor search direction from the subproblem solution. A poor search direction will result in a very small step size to improve the descent function. This results in many func-

tion calls because the step size determination requires may iterations with each iteration requiring a complete finite element evaluation. Because the step is small, a very small change in the design is the result. The subproblem then returns almost the same search direction, starting the loop over again. When a poor search direction was determined one of two things would happen: 1) The step size iteration maximum could be reached (this was at 25) terminating the algorithm, 2) The algorithm could make some very limited progress and then finally make a large move in the design space. The non-convergence problem is well illustrated by figures 10 and 16 where no value of R was able to produce a solution.

Two things were found to help for non-convergence in some cases. A routine was added to the basic algorithm to add more constraints to the potential constraint set (17:2194). This was done when the algorithm required more then 15 iterations to determine a step size. This enriches the subproblem. It had limited success. The convergence of $R = 10$ and $R = 50$ in figures 14, Case No. 6 was determined to have been helped by the above method. Different values of R were also shown to help. A larger value of R put greater weight on constraint violation correction and resulted most often in a different search direction avoiding the point in

Fig. 09 - Case No. 1

CSD - Three-Bar Truss

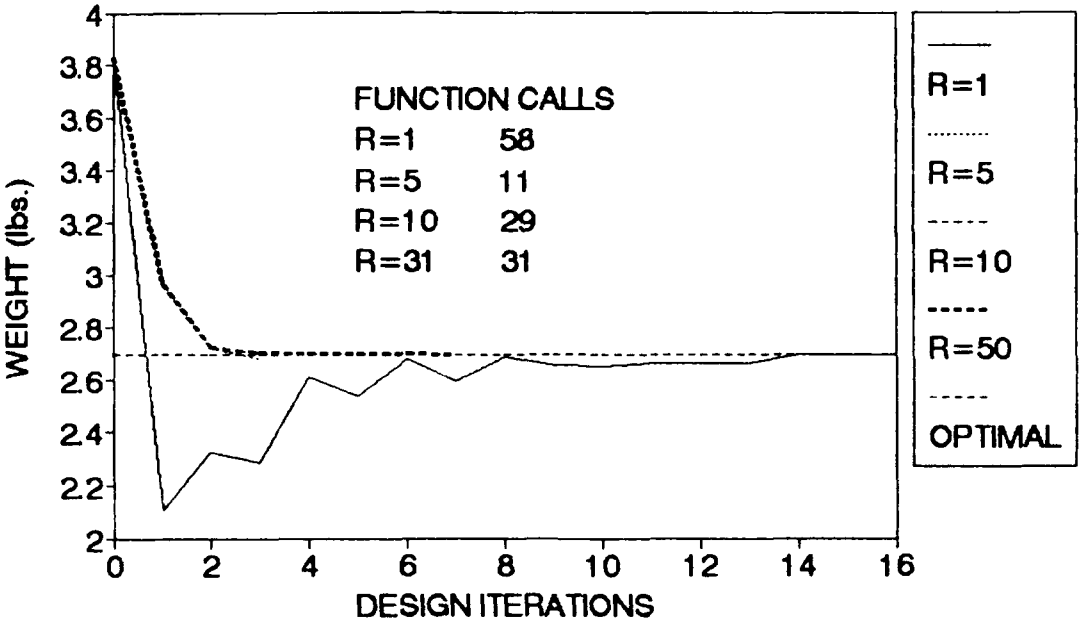


Fig. 10 - Case No. 2

CSD - Three-Bar Truss

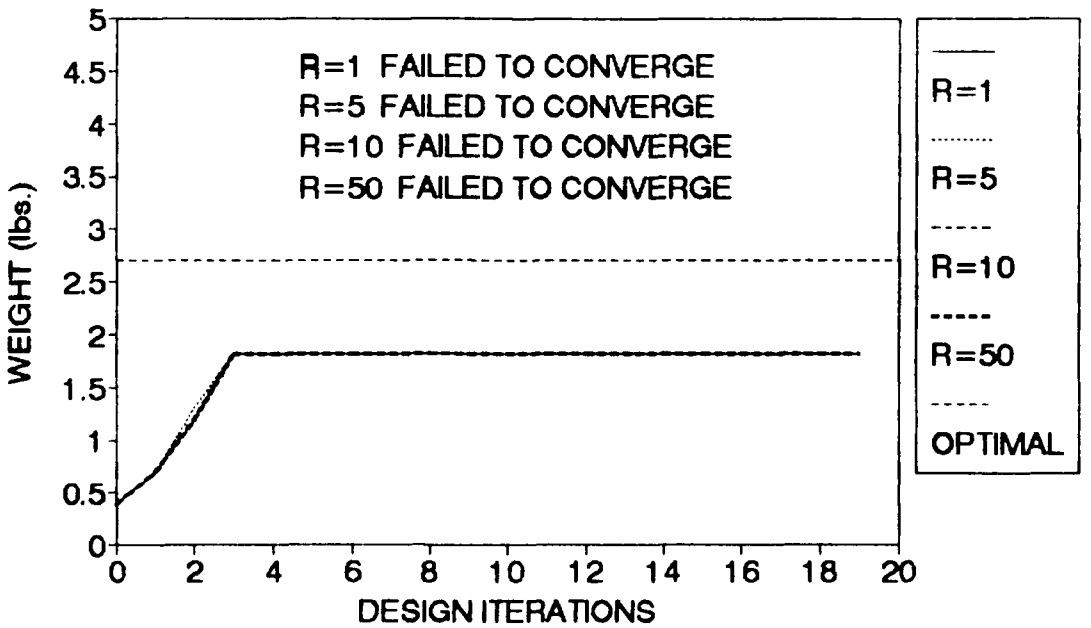


Fig. 11 - Case No. 3

CSD - Three-Bar Truss

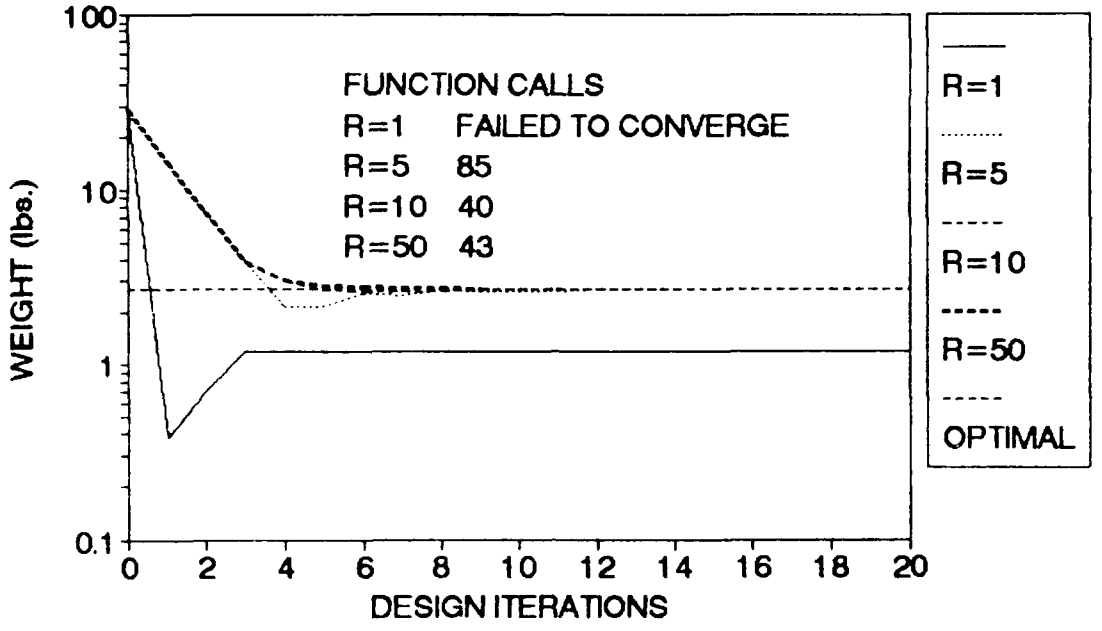


Fig. 12 - Case No. 4

CSD - Three-Bar Truss

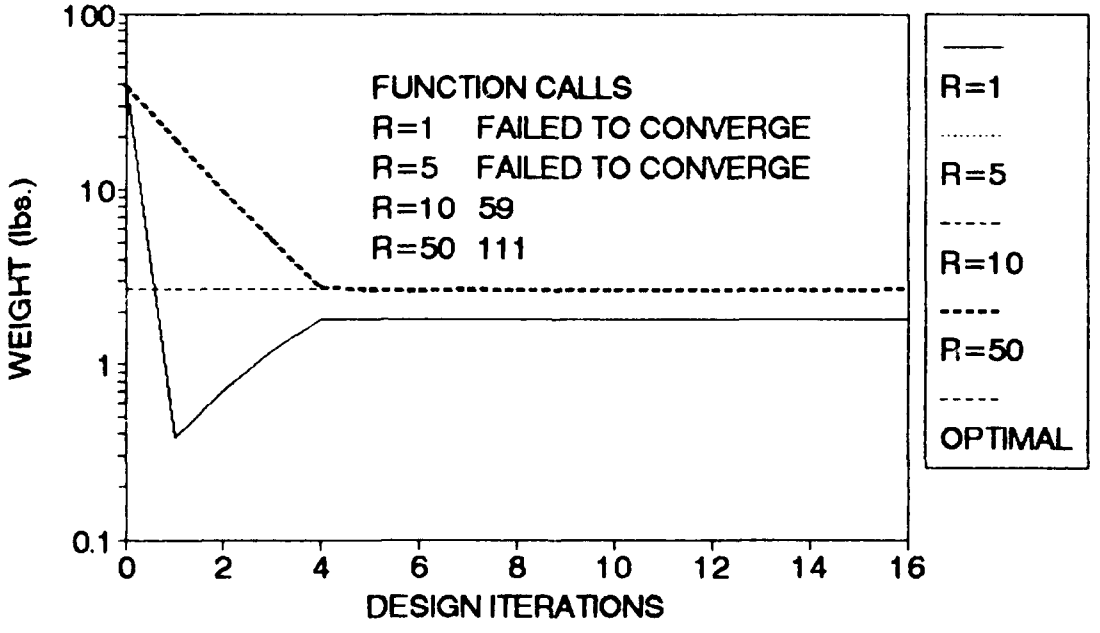


Fig. 13 - Case No. 5

CSD - Three-Bar Truss

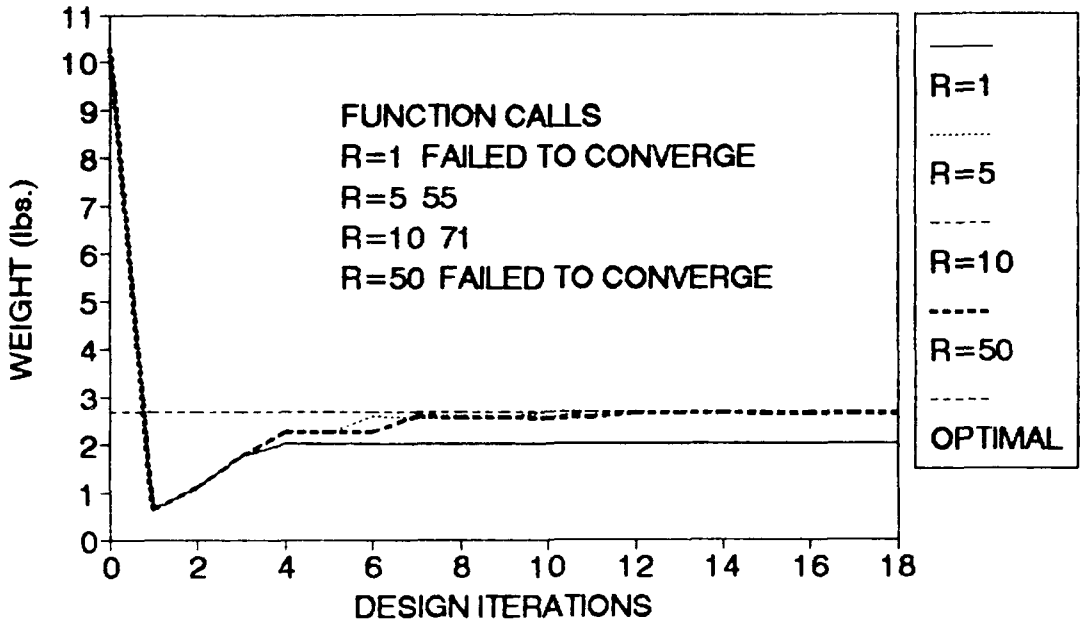


Fig. 14 - Case No. 6

CSD - Three-Bar Truss

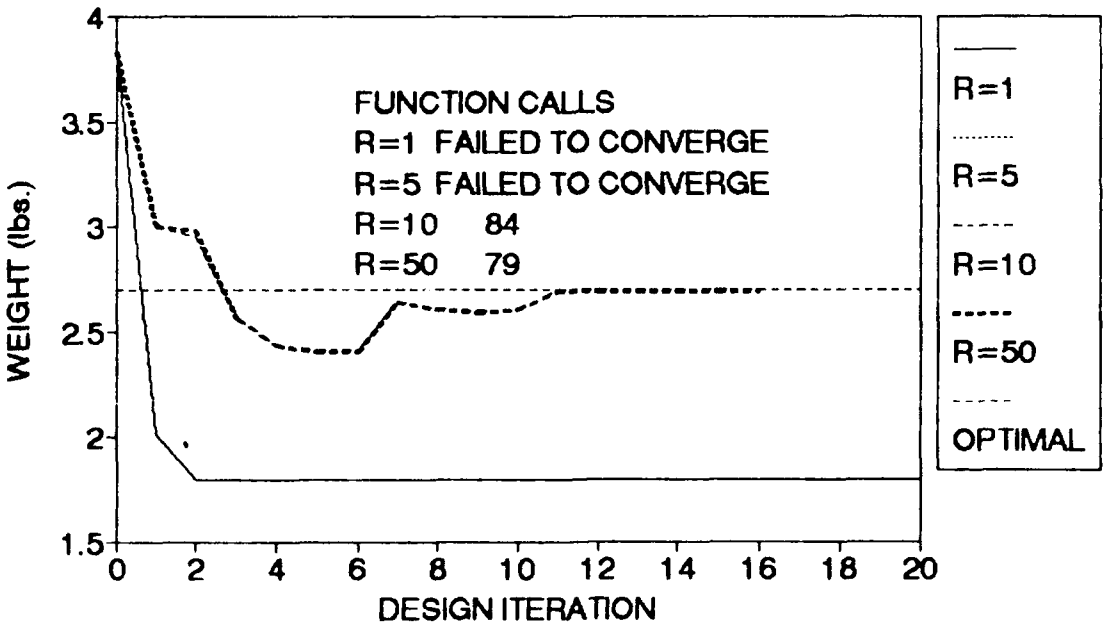


Figure 15. Case No. 7

CSD - Three-Bar Truss

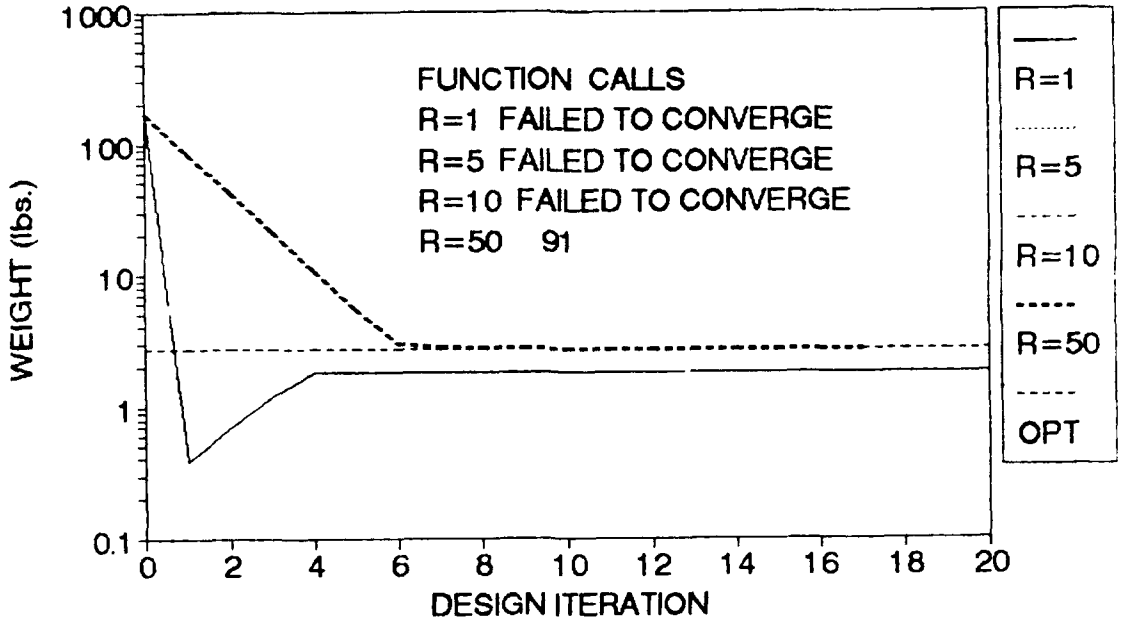


Figure 16. Case No. 8

CSD - *Three-Bar Truss*

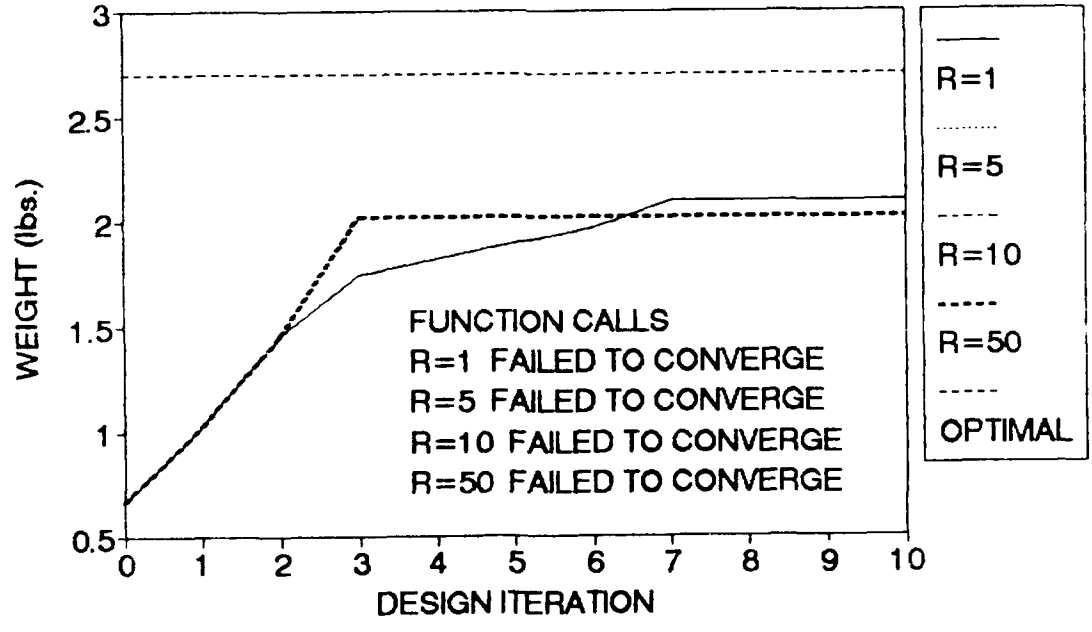


Figure 17. Case No. 9

CSD - Three-Bar Truss

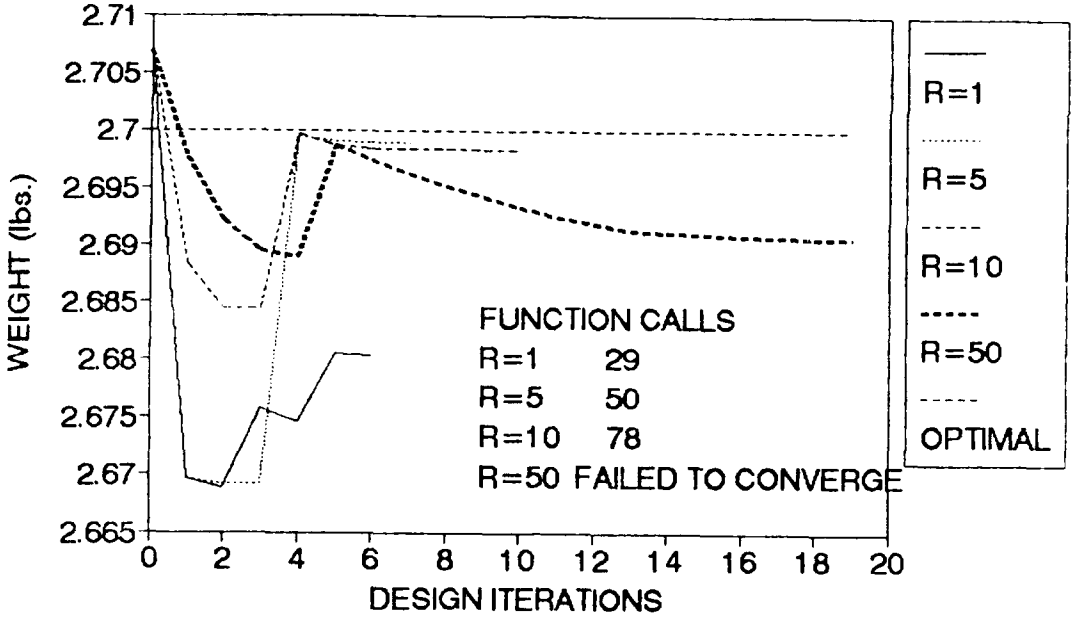
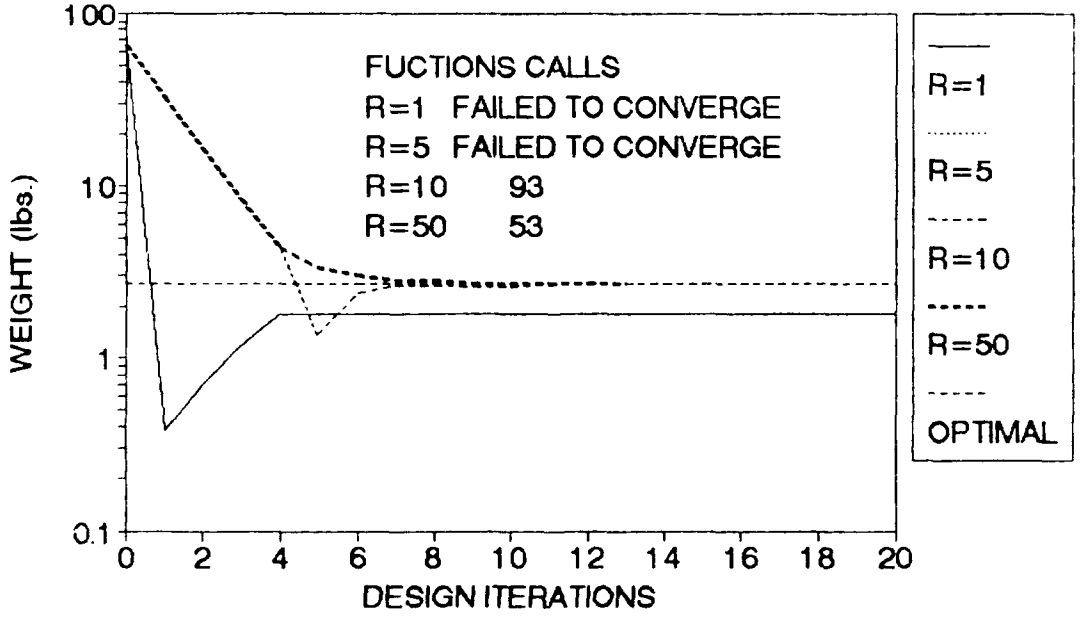


Figure 18. Case No. 10

CSD - Three-Bar Truss



the design space that resulted in the poor search direction. This is very clearly seen in figures 11, 12, 14, 15, and 18. In figures 13 and 17 $R = 50$ failed to converge but was very close to the optimal design.

The non-convergence problem also seems to be sensitive to the starting design. In figures 2 and 8 all values of R failed to converge to a solution. Both cases (2,8) where heavily violated in terms of constraints for both design variables. Here the value of R would have little effect in the initial path of the design search. The other case with heavy maximum violation, case 5, only one of the design variables was severely under designed. This can be seen from the starting weights of cases 2, and 8 as opposed to 5. The increased value of R was shown to be effective here.

No clear relationship was shown numerically for the value of R and the function calls required. A larger value of R should result in smaller movements in the design space hence resulting a larger number of function calls but this was not always the case. This is possibly the result of the poor search direction problem.

The poor search direction problem could possibly be taken care of through an added step of Hessian updating (Quasi-Newton Methods) (3:407). Hessian updating is an advanced feature of most algorithms of this type which includes higher-order information and helps provide a more efficient search direction.

The second major problem in implementing the CSD algorithm was the infeasible sub-problem. This problem was not encountered in the three-bar test problem but was the reason that meaningful results could not be obtained from the ten-bar problem. The infeasible sub-problem is very common with algorithms that use a Quadratic Programming subproblem (16:249). It is the result of incompatible linear constraints. A routine was added to the basic algorithm to delete a constraint from the potential constraint set when the subproblem was infeasible. This opens the design space. This technique did not help the solution process in the ten-bar case. Other possible methods to help correct this problem is to multiply the right side of the constraints by a number from .9 to .99 (12:196). This helps reduce the incompatibly between the constraints by relexing the constrants .

DISCUSSION OF PHASE II RESULTS

All four methods were applied to the three-bar test problem. Design iteration histories for all 10 starting cases are shown in figures 19 to 28. Final designs with the maximum constraint violation for each case are listed in Tables 4 and 5. All methods converged to a solution for all cases except for CSD in cases 2 and 8.

In all ten cases MSC/NASTRAN was shown to be the more conservative in its approach to the optimum. MSC/NASTRAN's descent was less steep then the other methods. This was the

result of having a more restricted movement in the design space because of always maintaining a feasible design in its search. MSC/NASTRAN showed very good accuracy as shown in tables 4 and 5. Starting design played a big factor in determining function calls and over all efficiency. Figures 21, 22, 23, 25, and 28 show that the further away MSC/NASTRAN is from the optimal design the more design cycles that are required. This is the result of the more conservative move limits placed on its design search.

The basic FSD solution is not the exact optimum for this problem. This can be seen in tables 4 and 5. For all 10 cases the FSD method converged to its exact FSD as expected. The number of iterations/function calls required for each case were fairly constant, compared to MSC/NASTRAN and CSD, showing good robustness for this problem.

FSD with scaling is shown to be the best method for this problem in terms of accuracy for a given amount of function calls. The method converged to a solution very close to the optimum in all but case 3 (table 4). The iterations required ranged from 2 to 4. Unlike the basic FSD, FSD with scaling increases its sensitivity to starting conditions.

The ten-bar results are shown in figures 29-33 with final designs in table 6. The results show That the FSD method performed very well in this problem. For

Figure 19. Case No. 1

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

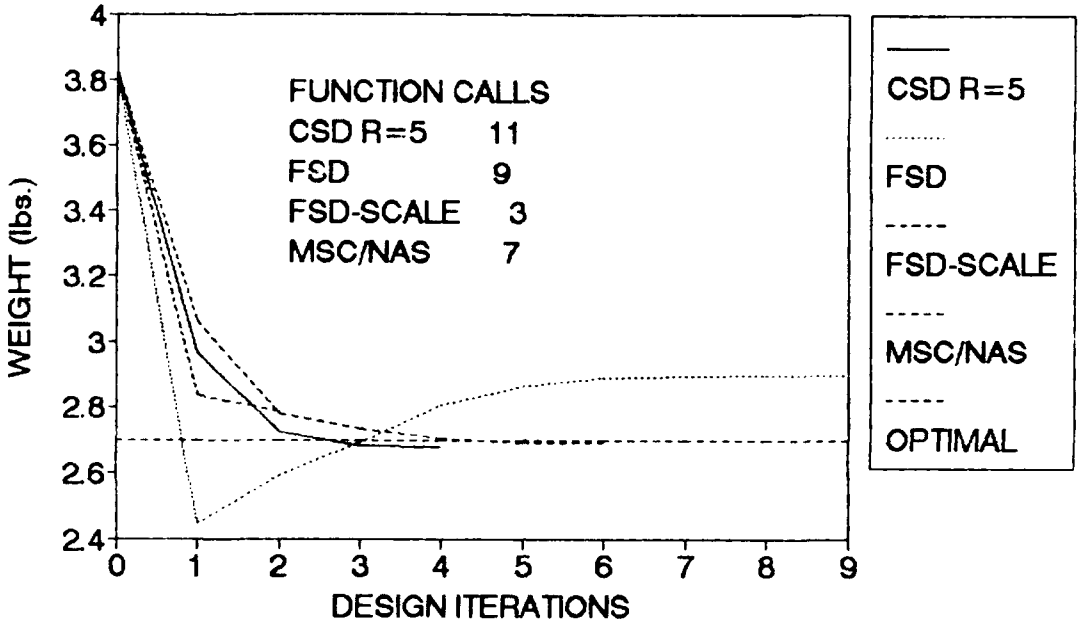


Fig. 20 - Case No. 2

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

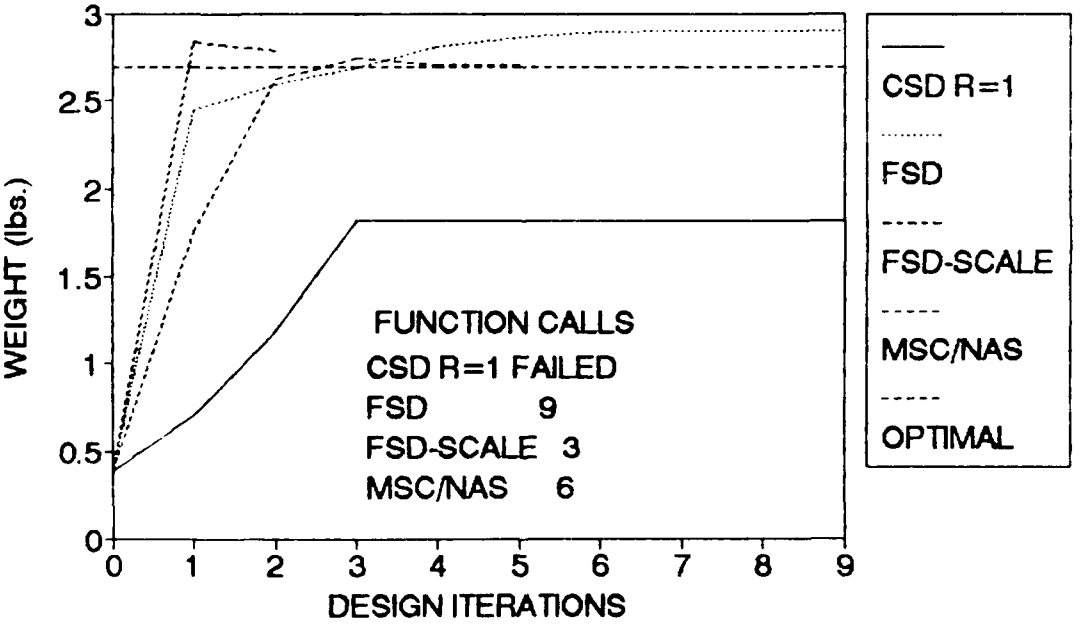


Figure 21. Case No. 3

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

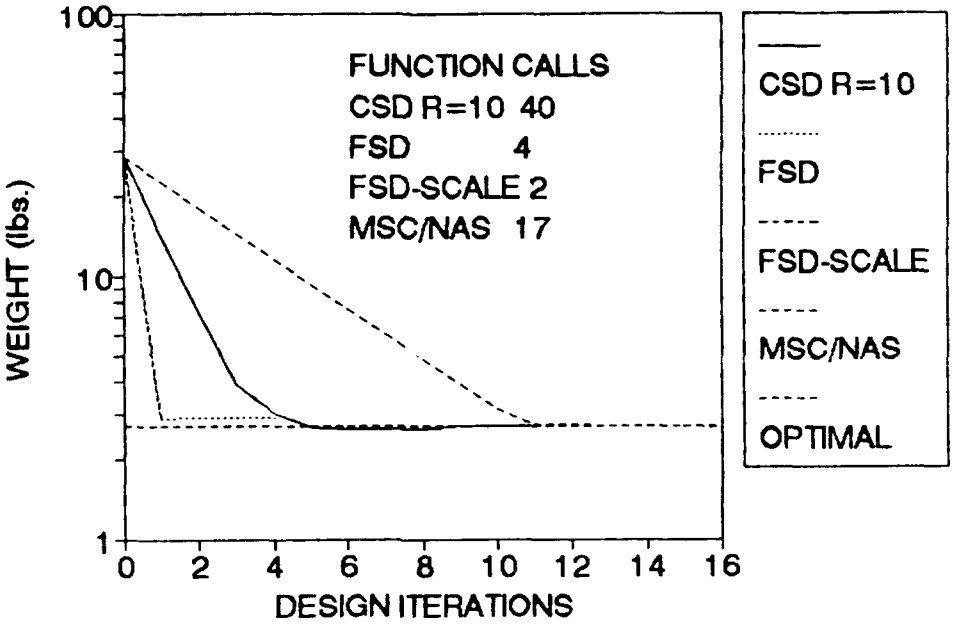


Figure 22. Case No. 4

CSD,FSD,MSC/NAS - Three Bar Truss

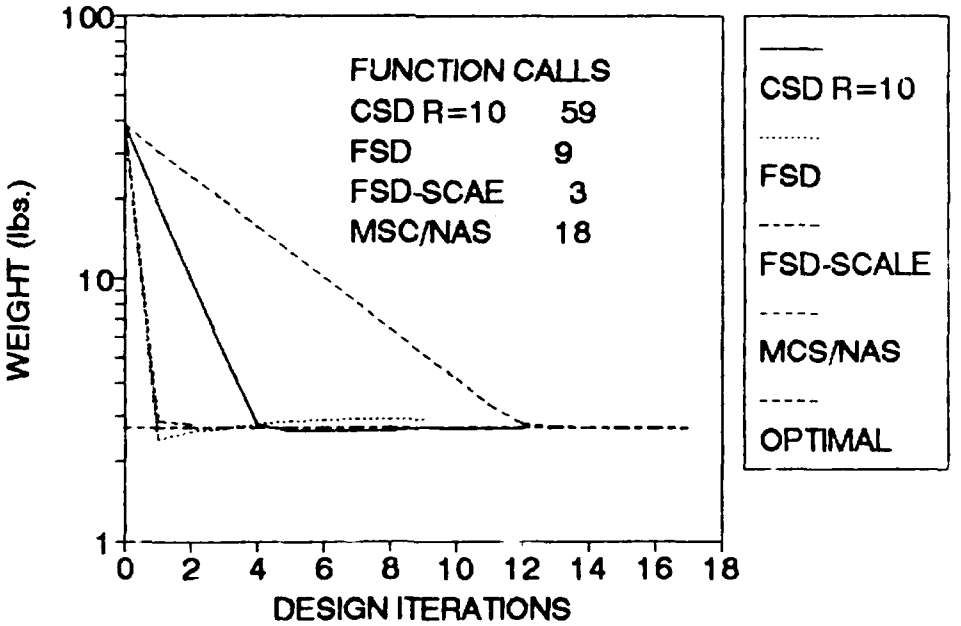


Fig. 23 - Case No. 5

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

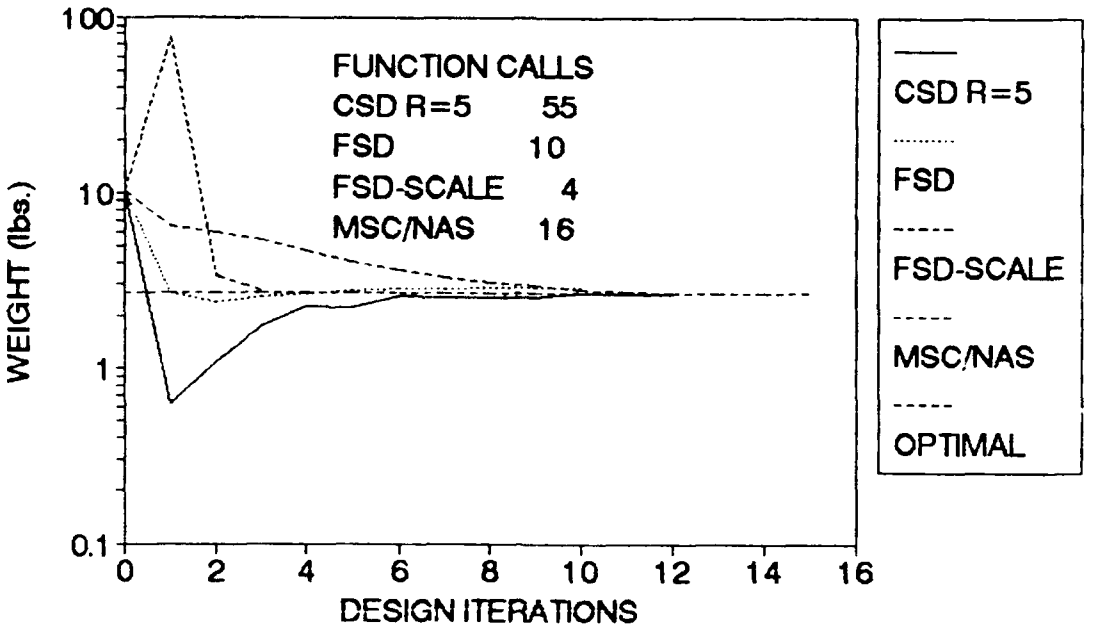


Figure 24. Case No. 6

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

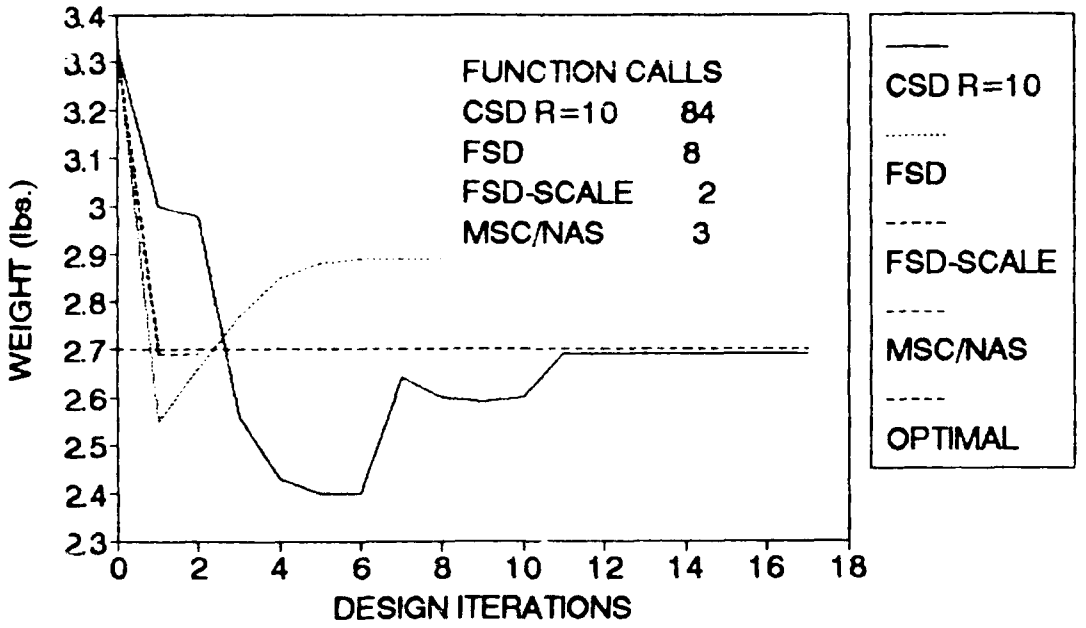


Figure 25. Case No. 7

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

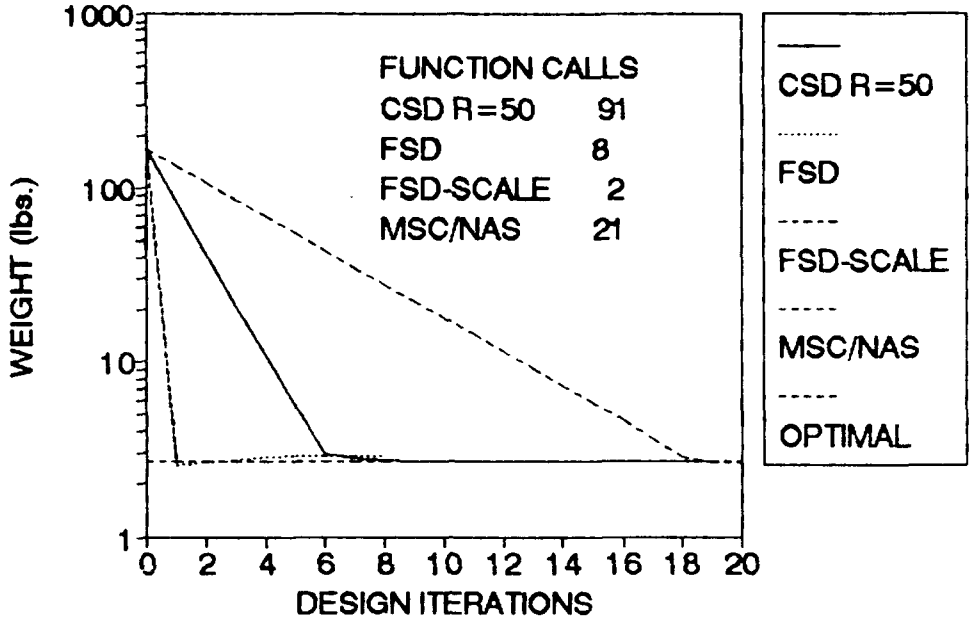


Figure 26. Case No. 8

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

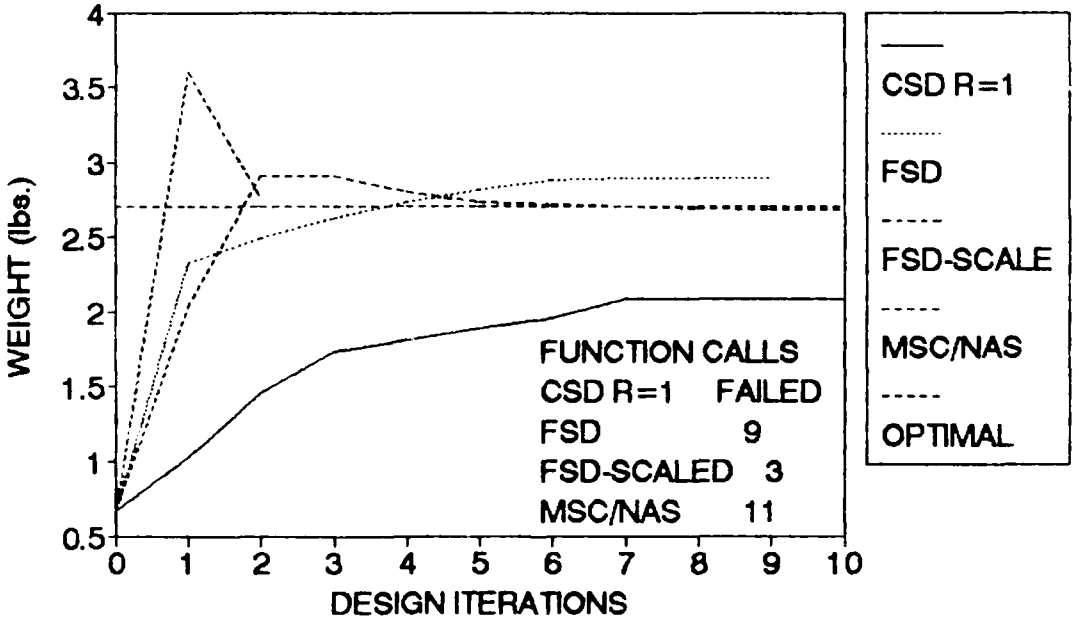


Figure 27. Case No. 9

CSD,FSD,MSC/NASTRAN - Three-Bar Truss

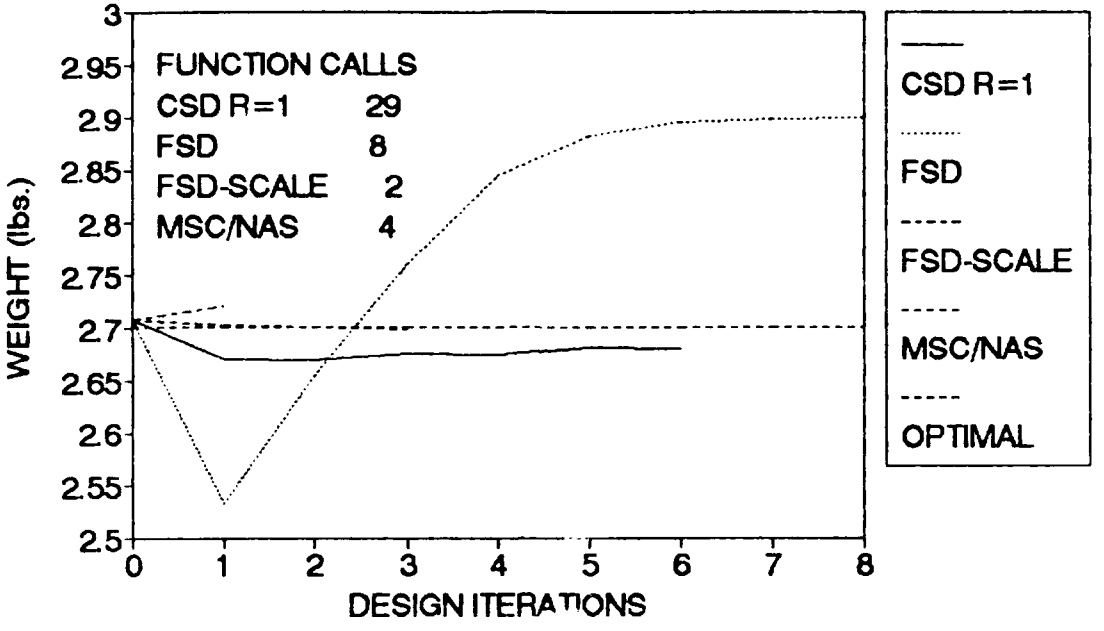
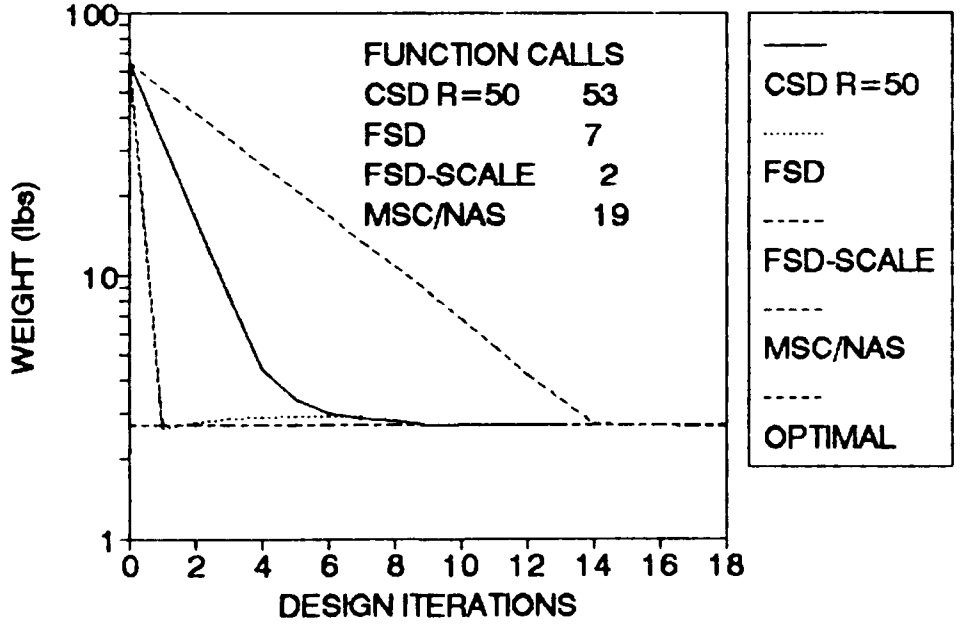


Figure 28. Case No. 10

CSD,FSD,MSC/NASTRAN - Three-Bar Truss



CASES	1	2	3	4	5
CSD	2.68/.0068	FAILED	2.69/.0019	2.69/.005	2.69/.004
FSD	2.90/0	2.90/0	2.90/0	2.90/0	2.90/0
FSD-SCA	2.79/0	2.79/0	2.88/0	2.79/0	2.78/0
MSC/NAS	2.69/.0026	2.71/0	2.69/.0046	2.69/.0046	2.69/.0046

TABLE 4. THREE-BAR TRUSS FINAL DESIGNS WEIGHT/MAX CONSTRAINT VIOLATION FOR CASES 1-5

CASES	6	7	8	9	10
CSD	2.69/.005	2.70/0	FAILED	2.68/.0073	2.70/0
FSD	2.90/0	2.90/0	2.90/0	2.90/0	2.90/0
FSD-SCA	2.71/0	2.71/0	2.76/0	2.72/0	2.71/0
MSC/NAS	2.69/.0045	2.69/.0034	2.69/.0049	2.69/.0049	2.60/.0049

TABLE 5. THREE-BAR TRUSS FINAL DESIGNS WEIGHT/MAX CONSTRAINT VIOLATIONS FOR CASES 6-10

this problem the FSD solution is very close to the optimal. In terms of over all accuracy it had the best performance. It also performed well in terms efficiency and function calls, compared to the other two methods in all five cases. FSD with scaling performed very well also, but fell short of the accuracy showed in the three-bar problem. This method required a few more iterations then the basic FSD method. Some interesting iteration histories were shown by FSD with scaling in the first three cases. The method spiked very high into the over designed region before converging down to a solution. It appears that when the starting designs are scaled for these cases the position of the design line on the critical constraint surface results in a very over design for the first iteration. FSD with scaling also showed very little sensitivity to starting designs in this problem.

MSC/NASTRAN was also effective in determining a solution to this problem but was shown to be very effected by an over designed starting condition. The number of function calls increased with the very over designed starting condition. This is shown in figures 32 and 33. The slow movement through the design space is again the result of move limits. Also observed is the method's convergence short of the optimum as shown by the final design in table 5.

Figure 29. Case No. 1

FSD and MSC/NASTRAN - Ten-Bar Truss

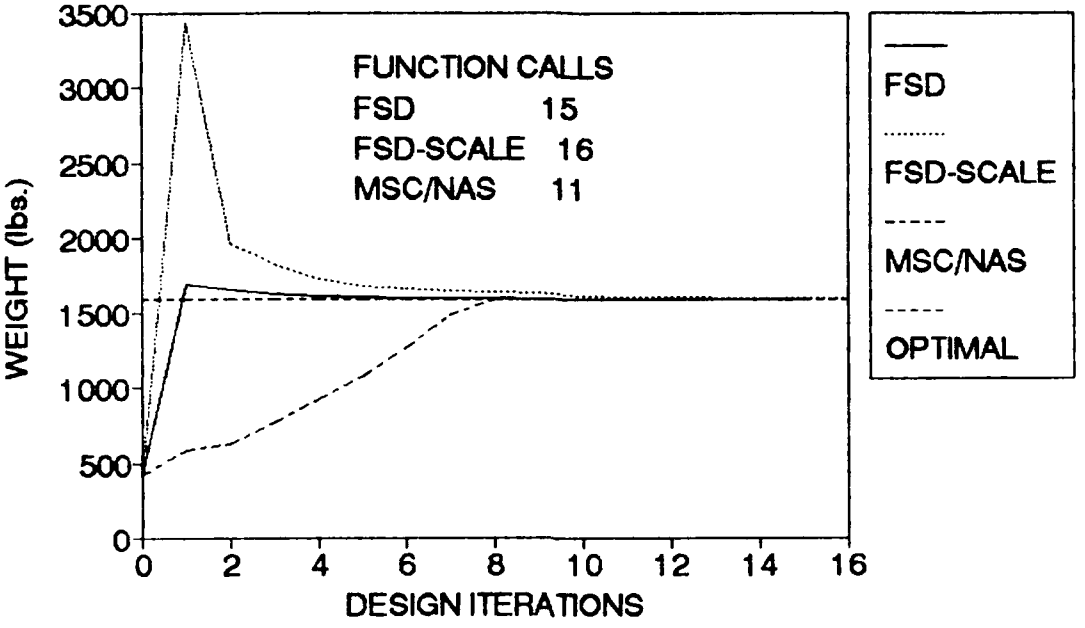


Figure 30. Case No. 2

FSD and MSC/NASTRAN - Ten-Bar Truss

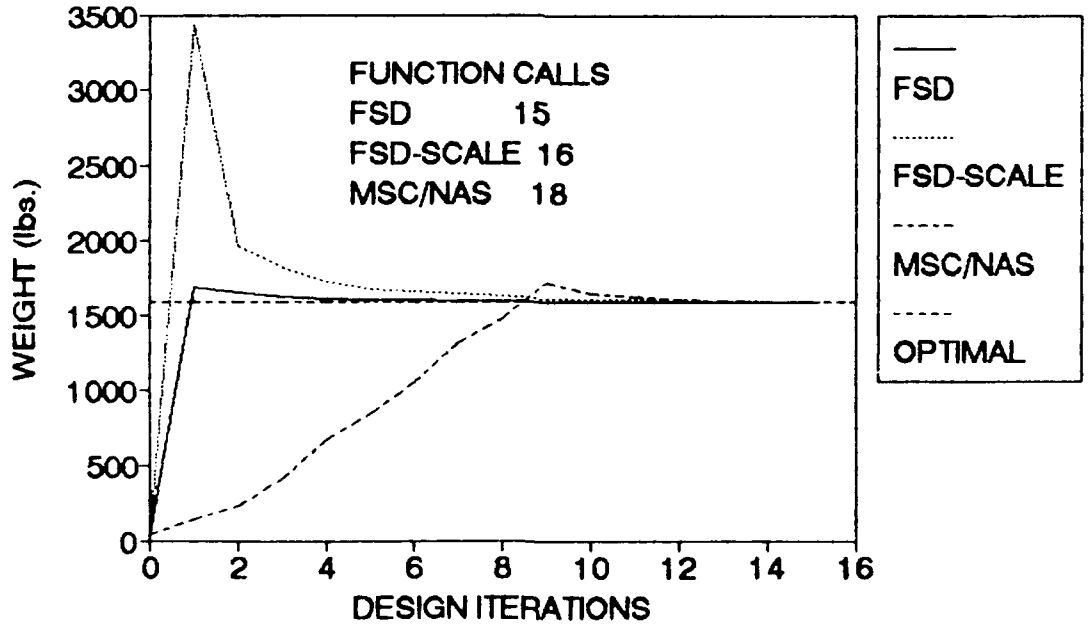


Figure 31. Case No. 3

FSD and MSC/NASTRAN - Ten-Bar Truss

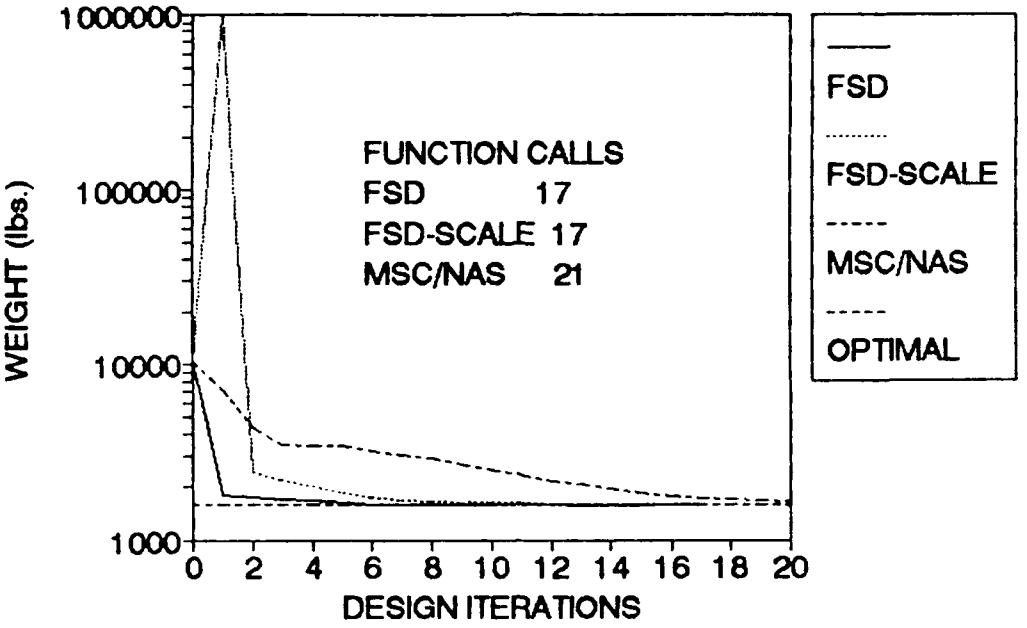


Figure 32. Case No. 4

FSD and MSC/NASTRAN - Ten-Bar Truss

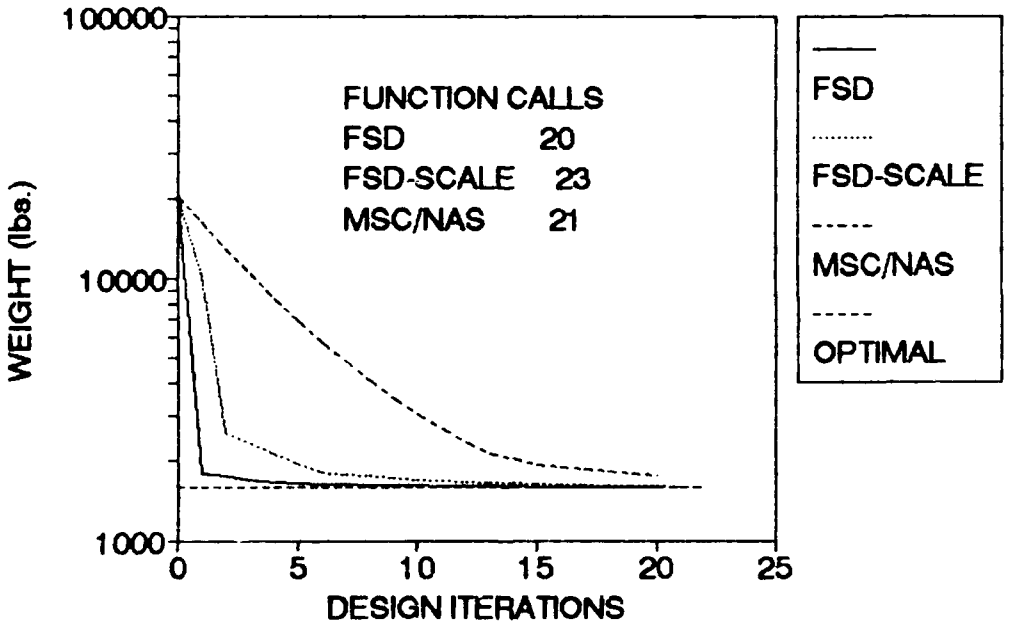
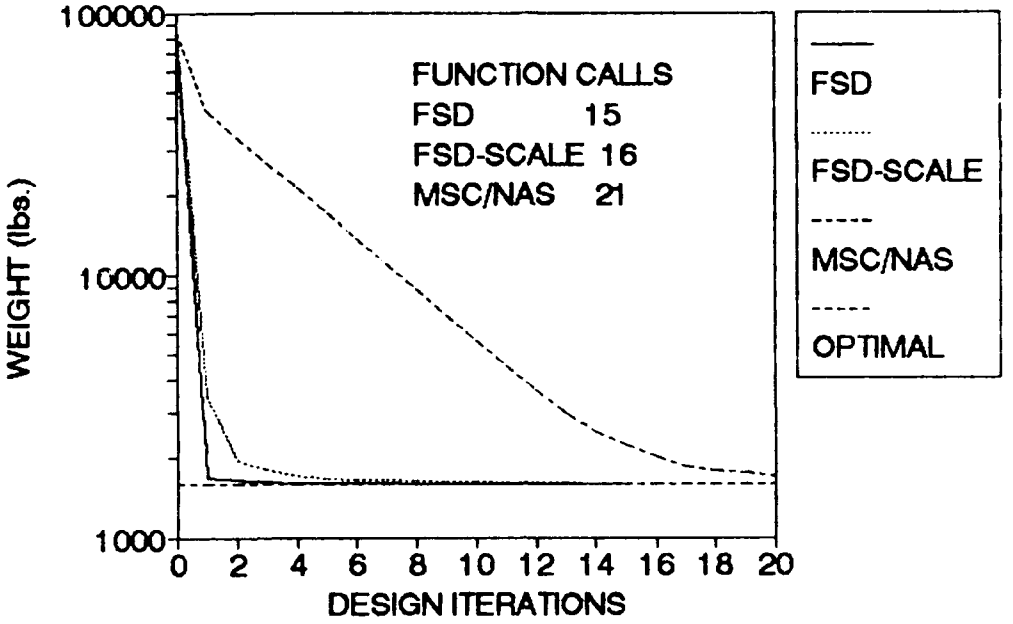


Figure 33. Case No. 5

FSD and NSC/NASTRAN - Ten-Bar Truss



CASES	1	2	3	4	5
FSD	1594.5/ .004	1594.5/ .004	1595.7/ .003	1591.6/ .008	1594.5 / .004
FSD-SCA	1599.1/ 0	1599.2/ 0	1599.1/ 0	1599.1/ 0	1599.1 / 0
MSC/NAS	1593.9/ .0016	1594.3/ .0016	1653.8/ .00088	1763/ .00061	1733/ 0

TABLE 6. TEN-BAR TRUSS FINAL DESIGNS WEIGHT/MAX CONSTRAINT VIOLATIONS

VII. CONCLUSIONS

Conclusions are drawn on two levels. The first level will be in terms of the results from the specific test problems with the second being a extrapolation to more complex problems. All four tested methods will be graded for accuracy, reliability, and efficiency.

CSD. In terms of accuracy CSD showed excellent results for the problems it could solve. Its reliability, as implemented is poor. It failed to solve some of the three-bar starting conditions and had no success with the ten-bar case. It was also very poor in efficiency. The inexact line search uses no approximations and requires a full finite-element evaluation for iteration in the step size determination. This coupled with poor search directions from the QP solver resulted a lot of function-calls.

FSD. FSD was very accurate for the ten-bar case but only fair in the three-bar case. Its reliability was excellent for both test cases. Of all the methods investigated it showed almost no variance between starting design and final solution. Its efficiency was also good although its convergence to a solution for the three-bar case was slow.

FSD-SCALE. FSD-SCALE was the best all round method for this set of problems. Its accuracy was reasonable good for both problems. Sensitivity to starting condition was shown in the three-bar problem but its reliability was still very

good. It had the best efficiency of any method in the three-bar problem converging in only two iterations for most of the starting cases. It also performed well for the ten-bar in terms of efficiency.

MSC/NASTRAN. Accuracy was very good for MSC/NASTRAN's version of Feasible Directions, although less accurate solutions were encountered in the over designed starting cases for the ten-bar problem. Its reliability was very good also. It converged to reasonable solutions for all starting cases applied to both problems, but was found to be sensitive to over designed starting conditions. The method was found to be efficient. The approximations used in the step size determination resulted in slower movement through the design space but only required one function call or finite-element evaluation per design cycle.

The biggest difference observed between the two direct methods and the two in-direct methods is their flexibility. Both CSD and MSC/NASTRAN (FD) have the ability to solve a variety of different problems with different kinds of constraints. The FSD methods are limited to problems with stress constraints. This is the result of a more general formulation of the problem by CSD and MSC/NASTRAN.

What is gained from the lack of flexibility though is simplicity. The FSD methods are easy to set up and understand and for specific problems can be very effective. But they also requires a greater understanding of the specific

problem by the designer. As stated earlier some structures may not have a FSD at all or may have more then one. The FSD method is only as good as the FSD for the structure is close to the optimal weight.

VIII. Suggestions and Recommendations

There are many possibilities for further work in this topic area. One possibility is further development of CSD with FRAME. CSD could be made to incorporate the advanced features of Hessian updating. Scaling between the objective function and the constraints also needs to be investigated. This could be a source of the poor search direction and infeasible subproblem concerns. FRAME could be updated to handle space trusses and/or beams and dynamic constraints such as natural frequency. This would allow for other constraint possibilities. These updates would create a manageable state-of-the-art Direct method which is on cutting edge of structural optimization research today.

Further work needs to done in looking at specific classes of structural problems in terms of optimization. Many of the Indirect-methods work best for specific types of problems. The possibilities of using various methods together could be investigated. For example using FSD to determine a starting design for CSD or FD to operate on. These results could aid in the development of a structural optimization data base.

Appendix

Instructions for Preparing Input Data for the CSD/FRAME Design System

GENERAL INFORMATION

The CSD/FRAME Design System uses two input files. One file inputs the basic structural data and is almost exactly the same as the normal input file for frame (10:359). The file inputs the optimization data.

The system outputs three files. The first file is the normal FRAME output file with all the finite element evaluations required by the problem run. The second file contains a complete history of the optimization process. Finally the third file is a summary of the optimization history.

The system is comprised of the following files: PROC.F (Contains the master program with all data processing subroutines), FRAME.F (Contains the basic FRAME program with modifications), APPROX.F (Contains subroutines to evaluate the constraint functions, form the potential constant set, and evaluate the gradients of objective and constraint equations, HQUAD.F (The QP solver), DESIGNC.F (Checks the design for convergence), and STEP.F (Determines the required step along the search direction).

INSTRUCTIONS FOR FRAME INPUT

Instructions for FRAME input are listed in Ref. 10 pg 359. The instructions should followed completely except for the following:

* In Card Set 2, NMATS must equal the number of members in the truss. Also NPROPS must equal 1.
* In Card Set 3, PROPS(JMATS,2) is left blank
* In Card Set 4, JELEM must equal the design variable no.

INSTRUCTIONS FOR OPTIMIZATION INPUT

* Card Set 1 (2I10)
  Cols 1-10 NDV (No. of design variables)
  Cols 11-20 NMBRS (No. of members)

* Card Set 2 (F15.5)
  Cols 1-15 PROPS(IELEM,2) (Initial cross-sectional area for a given member-must entered in element no. order-one card for each element)

* Card Set 3 (2F15.5)
  Cols 1-15 COMLIM (Compression limit)
  Cols 16-30 TENLIM (Tension limit)

* Card Set 4 (2F15.5)
  Cols 1-15 DISLIV (Displacement Limit-Vertical)
  Cols 16-30 DISLIH (Displacement Limit-Horizontal)
  Cols 31-45 DVLIM (Design variable limit)

* Card Set 5 (F15.5)
  Cols 1-15 WTDEN (Weight density of the material)

* Card Set 6 (I10)
  Cols 1-10 JMBRS(JDV) (Design variable for each member-one card for each member

* Card Set 7 (F15.5)
  Cols 1-15 E (Variable used in determining the potential constraint set)

* Card Set 8 (F15.5)
  Cols 1-15 R (Initial penalty parameter descent condition)

* Card Set 9 (F15.5)
  Cols 1-15 Gamma (Descent function parameter)

* Card Set 10 (2F15.5)
  Cols 1-15 EP1 (Convergence parameter search direction)
  Cols 16-30 EP2 (Convergence parameter for constraint violation)

* Card Set 11 (I10)
  Cols 1-10 IMAX (Max no. of step size iterations)

Bibliography

1. Kirsch, Uri. Optimum Structural Design. New York: McGraw-Hill Book Company, 1981.

2. Arora, Jasbir S. And Ashok D. Belegundu. "A Study of Mathematical Programming Methods for Structural Optimization Part I: Theory," International Journal for Numerical Methods in Engineering, 21: 1583-1599 (1985).

3. Arora, Jasbir S. Introduction to Optimum Design. New York: McGraw-Hill Book Company, 1989.

4. Venkayya, Vipperla B. "Structural Optimization: A Review and Some Recommendations," International Journal for Numerical Methods in Engineering, 13: 203-228 (1978).

5. Schmit, L.A., Jr. "Structural Optimization-Some Key Ideas and Insights," New Directions in Optimum Structural Design, edited by E. Atrek and others. New York: John Wiley & Sons, 1984.

6. Arora, Jasbir S. And Ashok D. Belegundu. "A Study of Mathematical Programming Methods for Structural Optimization Part II: Numerical Results," International Journal for Numerical Methods in Engineering, 21: 1601-1623 (1985).

7. Venkayya, V.B. And others. "Comparison of Optimality Criteria Algorithms for Minimum Weight Design of Structures," AIAA Journal, 17: 182-190 (February 1979).

8. Reklaits, G.V. And others. Engineering Optimization Method and Applications. New York: John Wiley & Sons, 1983.

9. Tseng, C.H. And J.S. Arora. "On Implementation of Computational Algorithms for Optimal Design 1: Preliminary Investigation,"International Journal for Numerical Methods in Engineering, 26: 1365-1382 (1988).

10. Hinton E. And D.R.J. Owen, An Introduction to Finite Element Computations. Swansea, U.K.: Pineridge Press Limited, 1979.

11. Press, William H. And others. Numerical Recipes, The Art of Scientific Computing. Cambridge: Cambridge University Press, 1986.

12. Vanderplaats, G.N. Numerical Optimization Techniques for Engineering Design with Applications. New York: McGraw-Hill, 1984.

13. Park, G.J. And J.S. Arora. "Role of Database Management in Design Optimization Systems," Proceedings of the 27th AIAA Structures, Structural Dynamics and Materials Conference. 620-629. San Antonio:AIAA, 1986.

14. Wolfe, P. "The Simplex Method for Quadratic Programming," Econometica, 27: 382-398 (March 1959).

15. Hadley, G. Nonlinear and Dynamic Programming. Reading, Ma: Addison-Wesley, 1964.

16. Gabriele, G.A. And T.J. Beltracchi. "A Investigation of Pshenichnyi's Recursive Quadratic Programming Method for Engineering Optimization," Journal of Mechanisms, Transmissions, and Automation in Design, 109: 248-256 (June 1987).

17. Thanedar, P.B. And others. "Performance of Some SQP Algorithms on Structural Design Problems," International Journal for Numerical Methods in Engineering, 23:2187-2203 (1986).

18. Haftka, R.T. And M.P. Kamat. Elements of Structural Optimization. Dordrecht, The Netherlands: Martinus Nijhoff, 1985.

19. Mitchell, A.G.M. "The Limits of Economy of Material in Framed Structures," Phil. Mag., 6: 589-597 (1904).

20. Venkayya, V.B. "Design of Optimum Structures," Computers and Structures, 1:265-309 (1971).

21. Miura, Hirokazu. MSC/NASTRAN, Handbook for Structural Optimization. Los Angeles: MacNeal-Schwendler Corp., 1988.

Vita

Captain Harry Hopkins III [redacted] He graduated from Half Hollow Hills High School in Dix Hills, New York in 1973 and attended the Academy of Aeronautics, graduating with an Associates Degree in Aircraft Design in 1976. He enlisted in USAF and served as a Personnel Technician at U.S. Air Force Academy. After being selected for the Airmans Education and Commissioning Program (AECP), he attended Texas A&M University and graduated in 1980 with a Bachelor of Science in Aerospace Engineering. Upon graduation he attended Officers Training School and received a commission in the USAF. His time as an officer began at Hanscom AFB, Mass. as an Aircraft Systems Manager for the E-4B, National Emergency Airborne Command Post (NEACP). During that time he earned a Master of Science in Systems Management from Western New England College. In 1985 he was selected as an Air Force Reserve Officers Training Corp (AFROTC) Instructor at Boston University. There is was the Commandant of Cadets, responsible for the training of over 100 Cadets until entering the School of Engineering, Air Force Institute of Technology, in July 1989.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	Master's Thesis

4. TITLE AND SUBTITLE

INVESTIGATION OF DIRECT AND INDIRECT OPTIMIZATION ALGORITHMS FOR AEROSPACE STRUCTURES

5. FUNDING NUMBERS

6. AUTHOR(S)

Harry Hopkins, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
WPAFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GA/ENY/90D-7

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for Public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This Study investigated the performance of direct and two indirect methods of structural optimization. A gradient calculator and overall design system was created with the finite element code FRAME. Three of the methods were employed using this system with the fourth being a commercially available code. The algorithms were applied to two different sized trusses using static constraints and were measured for accuracy, reliability, and efficiency. Results for the problems tested showed th direct methods were sensitive to starting designs and their performance depended on the proper selection of internal parameters. They were also shown to have a high degree of accuracy and the flexibility to handle different problems. The indirect methods showed that they were very effective when applied to specific problems and were simpler to implement and manage than direct methods, but lacked the flexibility to handle a variety of problems.

14. SUBJECT TERMS

Structural Optimization, Optimization, Truss Design
Mathematical Programing

15. NUMBER OF PAGES

91

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL